(1)

DEPARTMENT OF THE AIR FORCE

AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

(1)

AFIT/GE/ENG/88D-17

GRAPHICAL MAN-MACHINE INTERFACE FOR AN INTEGRATED EVALUATION ENVIRONMENT

THESIS

Jerry J. Kanski
Captain, USAF

AFIT/GE/ENG/88D-17

AFIT/GE/ENG/88D-17

GRAPHICAL MAN-MACHINE INTERFACE FOR AN INTEGRATED EVALUATION ENVIRONMENT

THESIS

Presented to the Faculty of the School of Engineering
of the Air Force Institute of Technology
Air University
In Partial Fulfillment of the
Requirements for the Degree of
Master of Science in Computer Engineering

Jerry J. Kanski, B.S.
Captain, USAF

December 1988

Preface

The purpose of study was to create an effective color graphical man-machine interface prototype. The interface prototype is needed to develop a common graphical interface for the various software resources used by AFIT researchers.

The interface prototype was successfully designed, implemented, and evaluated. Both the interface design and implementation met all of the research requirements and featured the abilities to be easily modified and expanded. The interface evaluation showed that the interface prototype was a solid success, and that the research into graphical man-machine interfaces should continue.

It takes more than one person to complete a thesis research effort. My faculty advisor, Capt M. Leahy, provided badly needed guidance and patience. Major P. Amburn taught me about graphics and gave me useful advice. I also received a lot of technical assistance and support from both Capt Leahy and Dan Zambon. Finally, the thesis and AFIT would have been impossible without my wife Vicki's support and patience. Thanks to the variety of help I received from these outstanding people, the AFIT thesis experience was rewarding and enjoyable.

Jerry J. Kanski

Table of Contents

List of Figures

List of Tables

Abstract

The purpose of this ~~research~~ thesis was to create an effective color graphical man-machine interface prototype for the various software resources used by AFIT researchers. The research was completed in four steps: (1) The general user requirements and specific interface requirements for the interface were determined; (2) An interface design which met all of the requirements determined in step 1 was created; (3) The design created in step 2 was implemented; AND (4) The man-machine interface was evaluated by a group of representative users.

The design effort resulted in a general purpose interface design that was highly adaptable and expandable. The design met all of the requirements determined in step 1, yet included the flexibility to meet future, as yet undetermined, requirements. The implementation of the interface design also featured adaptability and flexibility. The implementation combined a variety of graphical techniques to create a very powerful system. In addition, the resulting man-machine interface was evaluated by a group of AFIT engineering students and staff. The results of the evaluation indicated that the graphical man-machine interface prototype was a success.

GRAPHICAL MAN-MACHINE INTERFACE FOR AN INTEGRATED EVALUATION ENVIRONMENT

I. Introduction

Motivation

Modern computer systems used by the Air Force Institute of Technology for research are complicated and expensive. The process of designing hardware, developing software, testing systems, and integrating and interfacing the various systems with other systems and the real world is time consuming, error prone, and costly.

The AFIT researchers need an effective interface to their present computer resources to support in-house research and to aid in the evaluation and simulation of experimental systems. Researchers presently use a variety of sophisticated software resources. But the various software resources were designed for daily use by highly trained engineers, and their operation is neither simple nor obvious to the casual user. In addition, each of the software resources uses its own unique man-machine interface. A single, common, graphics-based interface to the various research software resources would improve the

research capabilities and greatly reduce the learning time of inexperienced operators, such as students.

Objective

The objective of this research effort was to create an operating prototype for an effective graphical man-machine interface for the various software resources presently used by AFIT research groups.

Problem Statement

Creating a graphics-based man-machine interface which will operate with a wide variety of applications and also effectively interface with human operators is a large and complex project. The first step in such a project is to create an initial prototype of the interface. But a prototype for a man-machine interface is in itself a large project with many technical problems. One problem is to develop an effective design that will meet the many interface requirements. Another problem is to show that the design can be implemented with the resources available.

A major challenge involved in implementing a graphics-based man-machine interface prototype is the effective management of the interface graphics features on the computer display. An interface is useless if the software resources, hardware resources, and screen management techniques will not meet the demands of an interface design requiring high-speed screen manipulations. An interface

is also useless if the interface design and implementation do not meet the requirements of the end users of the system.

The research completed for this thesis is divided into two parts. The first part concerned the design of a graphical man-machine interface prototype. The second part concerned the development of the screen management techniques needed to implement the interface prototype design. Both parts of the research are described in the following two sections. Following the descriptions is a section outlining the graphical man-machine interface considerations which were beyond the scope of this thesis effort.

Part one. The first part of the research consisted of a design for a graphical man-machine interface prototype which would meet the basic interface requirements of the AFIT software resources. The main design emphasis was on control flow, basic screen configurations, and basic man-machine interface considerations such as error handling, user skill levels, and feedback. The design did not include implementation aspects or functions required to meet specific software resource requirements.

Part two. The second part of the research consisted of the development of the screen management techniques needed to successfully implement the graphical portion of the man-machine interface prototype design. The techniques

were integrated with the basic control flow and displays of the design to demonstrate the prototype screen management operations.

Areas not Addressed. The thesis effort did not deal with any of the following man-machine interface considerations:

1. Software speed enhancements (efficiency).
2. Specific data input or output methods required by the application programs.
3. Data manipulations (other than data used for the screen management and specific interface displays).
4. File manipulations.
5. Color considerations in computer displays.
6. Aesthetics in computer displays.

Method of Approach

Four major objectives were reached during the research effort. The first objective was the determination of the basic interface requirements. The second objective was the design of an interface prototype which met the determined requirements. The third objective consisted of successfully implementing the complete design. The third objective actually contained two parts. The first part consisted of implementing the screen management techniques required by the interface design. The second part was the integration of the screen management techniques with the control flow and screens of the interface design. The

fourth and final objective was the evaluation of the operation of the interface. Descriptions of the four objectives are in the following paragraphs.

Requirements Analysis. The following six steps were completed to determine the initial requirements for the research project:

1. Specific user needs and preferences were determined from the literature search described in Chapter 2.

2. The use and operation of modern graphical man-machine interface techniques was determined from the literature search.

3. The basic operation of a GPX graphics workstation was tested.

4. The basic operation of a representative AFIT research program was tested.

5. The capabilities of the GPX workstation were determined.

6. The hardware and software requirements for the screen management techniques needed for the interface design were determined.

Project design. All three of the required milestones were reached during the design phase of the thesis research. The three milestones were:

1. The overall requirements for the man-machine interface were determined from the individual requirements found above.

2. The graphics screen management techniques which were needed to meet the requirements of the man-machine interface prototype were selected.

3. The graphical man-machine interface prototype was designed.

Implementation. The following technical operations were completed to implement the project design:

1. A program which duplicated the interface design control flow was written and tested.
2. Stubs for the graphics operations were added to the control flow program and tested.
3. Two graphics software packages, GKS and the DEC UIS graphics, were tested to determine if they would meet the interface requirements.
4. Basic windowing operations were implemented.
5. Screen layouts were implemented and tested.
6. Screen graphics (windowing operations and screen layouts) were incorporated into the graphics operations software.
7. Text and color were added to the graphics.
8. Mouse inputs were included to realistically test the interface operation.
9. Portions of the interface structure were configured to display the features supported by the design.

Interface evaluation. The operation of the basic design and the screen management techniques were evaluated by a variety of AFIT students and staff using a modified version of a CAD-Tool Human-Computer Interface Evaluation developed at AFIT (10). The Interface Evaluation was modified to reflect the basic purpose of the graphical man-machine interface prototype and to minimize the Interface Evaluation's emphasis on CAD tools.

Contribution and Summary of Results

All of the complex technical steps required to create

an operating, effective prototype graphical man-machine interface were successfully completed. In addition, the prototype man-machine interface was evaluated, with outstanding results, by a group of representative end users.

Milestone 1, an interface design. A highly adaptable and expandable general-purpose design was created. The design meets all of the general user and application requirements of a man-machine interface, yet provides the flexibility to meet future requirements. In addition, the interface was not just designed with the user in mind, but was created specifically from the users point of view.

Milestone 2, implementing the design. The complete interface design described in step 1 was successfully implemented during the research effort. The complex screen management techniques needed to implement the design were developed and integrated with the design control flow to create a fast and smooth color graphical man-machine interface.

Milestone 3, interface evaluation. As part of the thesis effort, a number of AFIT students and staff evaluated the prototype graphical man-machine interface. The evaluation results indicate the requirements, design, and implementation phases of the interface creation were all highly successful. The results of the evaluation were

used to identify areas requiring further research and will be used to improve the interface.

Summary of results. This thesis effort did not result in just an untested paper product. An interface design must be successfully implemented before it is possible to accurately test and evaluate the interface design. During this research, an effective color graphical man-machine interface prototype was designed, implemented, tested, and evaluated. The interface program is a smoothly working prototype which can be modified and improved upon as the basis of a very promising research effort to create a graphical man-machine interface which will successfully support a variety of research programs and users.

Organization

This thesis contains six chapters and four appendices. Chapter II is a survey of a variety of areas which directly relate to the creation of a color graphical man-machine interface. Chapter III contains the sections concerning the interface design. Chapter IV deals with the system implementation, and describes the implementation of the various parts of the complete design described in chapter III. The code developed during the implementation process is contained in an AFIT Robotic Systems Laboratory Technical Report (16). Chapter V covers the interface evaluation portion of the thesis effort. And in the last

chapter, results, conclusions, and recommendations are covered. Appendix A contains the psuedocode developed as part of the design effort. Appendix B contains a complete description of the code written for the implementation phase. Appendix C is a copy of the modified CAD-Tool Human-Computer Interface Evaluation. The last appendix, Appendix D, is the detailed results of the interface evaluation.

II. Literature Review

Overview

The creation of a successful man-machine interface requires the combination of knowledge from a wide variety of areas. To obtain that knowledge, five separate areas concerning graphical man-machine interfaces were researched before the design phase of the thesis effort began. The areas are; graphical dialog techniques, interface to user interaction, graphical interface design and implementation, graphical interface applications and examples, and one representative research application program. The five areas are reviewed in the following sections.

Graphical Dialog Techniques

Many graphical techniques are presently used by graphical man-machine interfaces. The six techniques most commonly used by successful graphical man-machine interfaces are menus, icons, windows, dynamic manipulation, color, and animation. Basic descriptions, including strengths and weaknesses, of the various techniques are contained in this section.

Menus. A menu consists of a list of text strings or graphical symbols which represent the computer functions

available to the user. The user normally selects one of the options displayed by the menu by using a pointing/ selection device, such as a mouse (6:206; 30:456).

There are two basic types of menus. The first type is the static menu. A static menu is displayed constantly during a given portion of a program and the user can select options from the menu at any time. The main drawback of static menus is the need to continually use part of the available screen area (27:73). The second type of menu is the dynamic menu. Dynamic menus appear on the screen when requested by the user and only stay on the screen until a selection is made or until the original request is cancelled. Dynamic menus can be selected at any time by the user, and only use screen space when needed. The main disadvantage of dynamic menus is that they require more user operations to make a selection (4:67).

Icons. An icon is a small graphical symbol for a computer function. A widely known example of the use of icons is the operating system for the Apple Macintosh, which uses icons for files, disk drives, and printers (42:30-31). To use an icon, the user selects the icon with a pointing/selection device and the program executes the function the icon represents. Icons are mostly used on personal computers and in specialized graphical application areas, such as draw, draft, and paint programs. Icons are visually more distinctive than text alone, and can

represent a lot of information in a small area of the screen (28:233). A drawback of icons is a large variety of icons can confuse the user, as in the case of some CAD (computer aided design) programs which use over 5000 individual icons (22:55).

Windows. A window is a portion of the screen dedicated to one function or to one program. In its simplest form a window is just the entire undivided computer screen. But the value of windows lies in the display of multiple windows at once (34:107). The user can then view multiple forms of data simultaneously, operate multiple tasks concurrently, or any combination of the two. Windows can be configured and manipulated to match the needs and preferences of the user. Windows are limited by screen size, screen resolution, and system processing capabilities. Also, too many windows can overcrowd the screen and force the user to spend excessive time on window manipulations (21:232-233).

Dynamic manipulation. Graphical dynamic manipulations are operations which use the pointing/selecting device to alter the graphical display on the screen or the contents of data files (25:51-52). Dynamic manipulations are used heavily for drawing, drafting, and paint programs, where the pointing/selecting device is used to graphically create and modify data (7:70; 15:36-37). In another form, dynamic manipulations can be used to move "controls" on the

computer screen which change the display screen or the control values of a real-time system (2:345-346). The main advantage of dynamic manipulations is that the user sees a graphical representation of the result of his/her inputs (20:415-416). A disadvantage of dynamic manipulations is the heavy use of screen area and computer resources required. The heavy CPU demands of dynamic manipulations can slow system response time and frustrate users (4:67).

Color. Using color increases the amount of information that can be conveyed to the user. The main uses of color on graphical interfaces are to catch the users attention, to signify a change in status, or as part of the actual representation of data. An advantage of the use of color is the display colors can be tailored to reduce eye strain and to make a display more attractive (12:46-47). A disadvantage of the use of color is the high overhead color usage imposes on the host system (19:141).

Animation. Animation is the use of movement on the computer display. Animations can range from simple implementations, such as a second hand rotating on an analog clock, to very complicated implementations using multiple objects, complex paths of motion, and changing colors. In user interfaces, animation is often used to dynamically update graphical representations of display devices or to highlight small changes on the screen which

otherwise might be missed by the user. Animation can provide even more information to a computer display, but animation requires a lot of programming effort and computer power (21:231).

Interface to User Interaction

The interaction between the user and the interface is very important. Two areas which directly affect the user/interface interaction, user needs and user preferences, are covered in this section.

User needs. The needs of the end users must be considered to create an effective man-machine interface. Five factors concerning user needs are especially important when dealing with robotics researchers and robotics research software resources. The five factors, user skill level, application relevance, feedback, error handling, and color are discussed in the following paragraphs (23:156).

User skill level. A man-machine interface should be tailored to the skill levels of the users. To do this, different dialog techniques are required for users with different skill levels (37:12). The interface should be configurable or flexible enough to handle the range of skill levels of the expected users. If the interface fails this requirement, users will be forced to work at skill levels different from their experience and training (4:67).

Application relevance. A man-machine interface should reflect the application program (12:46). The interface structure, displays, command options, dialogs, and syntax should be appropriate for the application program and the environment. The interface objects and functions should be meaningful, in the context of the application, and should provide the user with clear and consistent options (23:156).

Feedback. Users need fast and meaningful feedback to effectively interact with a man-machine interface. The interface should acknowledge user inputs and indicate to the user how the interface interpreted the input (40:239). Effective feedback can help lead a user through multiple selections, while bad feedback can confuse the user resulting in errors and user frustration (7:71; 29:238).

Error handling. A man-machine interface can help to minimize the chance of a user making an error. If an error is made, the user should be notified as soon as possible and given a chance to recover from the error. Error recovery is especially important when the user makes a destructive error, like an accidental file delete (23:156). An interface should not be sensitive to user errors, and should not react to an error by aborting the program (40:239).

Color. Color must be used carefully in a man-machine interface. A user should not have to remember the meanings of colors, so the number of colors used to denote meanings should be four or less (21:228-231). If colors are used to denote selections, standard meanings, such as red for stop, should be assigned. The excessive or incorrect use of color just confuses users (21:228-231).

User preferences. Users vary widely in their preferences concerning man-machine interfaces. But many preferences are common among most users. Two areas of preferences which directly affect the creation of an effective man-machine interface are user skill levels and graphical displays, both of which are discussed here.

User skill levels. The level of skill of users greatly affects what they want from a man-machine interface. Novices like menus, icons, and a lot of feedback and help from the system. A novice prefers to be led through an interface. But experienced users do not like the time and overhead associated with a complex graphical interface. Experienced users knows where in the system they want to go and what they want to accomplish. The experienced user prefers a simple interface, an uncluttered screen, quick commands, and functional shortcuts (4:67; 18:39; 31:31). Users want help available when they need it, and many users want configurable interfaces (9:65). An interface that meets these user

preferences must be flexible, responsive, configurable, and intelligent (18:39).

Graphical display. Users have conflicting preferences concerning the actual displays used with man-machine interfaces (12:46-47). For ease of use, users prefer fast system response and uncluttered screens with a limited number of colors. Users prefer simple graphics with clear and simple controls, and they like colors used for selections to be bright and easy to see against the background and other features (21:231). But on the aesthetic side, users like complex, busy displays with attractive color schemes. They prefer displays with low or medium contrast with blues and greens in pleasing combinations (12:47). A graphical man-machine interface must balance the conflicting preferences to be effective yet not annoying to the user (21:231).

On-line help. An effective man-machine interface should provide the user with quality help (13:48). There are four areas related to help which need to be addressed for an effective user help facility. The areas are; user considerations, commands and options, organization and navigation, and help about help. Each of these four areas is discussed in this section.

User considerations. Users at all levels of experience sometimes need help. An interface designed for users at different skill levels should provide different

levels of help (4:67). Help should be relevant to what the user is doing. A help facility should not bog down a user with help functions he did not request (9:65-66). Also, the quality of the help is more important to users than how the help is delivered (1:24).

Commands and options. A help facility should clearly tell the user what his options are and how to select them. If keyboard commands are allowed, the user should be given rules and examples of the syntax used for each command. And the help facility for given commands and options should be available at the points in the program where the user may need the help (9:66).

Organization and navigation. An effective help facility should show the user the application's organization and where he presently is in the organization (13:48). Giving the user location information helps him navigate through the application faster and helps him understand the operation of the application. The result is a quicker learning process and a more effective user (14:68; 30:13).

Help about help. A help facility is useless if a user does not know how to use it. Therefore, an effective help facility should help the user use the help facility. The capability is especially important to new users or if the help facility is complicated and has multiple options for different levels of help (9:66; 13:48-49).

Graphical Interface Design and Implementation

An effective graphical man-machine interface cannot be successfully designed and implemented without taking four areas into account. The areas are; user considerations, host system considerations, design structure, and implementation considerations.

User considerations. The design of a man-machine interface should start from the user's point of view (40:243). One goal of an interface is to make the user's job as easy as possible. Another goal is to give the user what he wants (23:156). To meet the goals an interface should be tailored for the needs and preferences of the user as well as for the requirements of the application. If an interface does not satisfy the user, the result will be a poor and seldom used product no matter how good the application program is (32:50-52).

Host system considerations. A graphical man-machine interface can use fifty percent or more of the code of a finished software system (3:30). The special requirements of color graphics add even more to the load on the host system (19:141). These facts mean the limitations of the host system must be considered by the interface designer. Some graphical techniques will be beyond the capabilities of the host system, and other techniques may slow the system almost to a standstill (41:41).

Design structure. Four approaches, tack-on, intuitive/empirical, formal, and conversational, have been used in the past for man-machine interfaces (39:11). Many present interface designs, partially for reasons mentioned in the application considerations section, use modular approaches (32:229). Many other reasons, all from modern software design practices, also support the modular approach.

Man-machine interfaces are complex and have a wide variety of requirements and functions (40:237-239). Present software practices, such as portability, standardized graphic protocols, and normalized devices almost demand a modular design (17:97). Many of the present modular designs use one of two structures. The first structure consists of a standard main program with a graphics subroutine package (36:243). The second structure, based on Seeheim's model of user interfaces, divides the interface into three components. The first component is a presentation component which handles the screen management, information display, input devices, interaction techniques, and lexical feedback. The second component is a dialog controller which handles the dialog between the user and the application. And the third component is an application interface module which deals with the interface's view of the application and handles the specific communication between the interface and the application (8:205-206).

Both design structures support a modular graphical interface design.

Implementation considerations. The characteristics of a graphical man-machine interface require two areas to be considered to successfully implement an interface. The two areas are concurrent interactive devices and prototyping.

Concurrent interactive devices. Graphical man-machine interfaces often use a variety of concurrent I/O devices. Dealing with multiple concurrent devices, such as mice, keyboards, and graphical output is a difficult task (26:249). Even with libraries of interaction functions, concurrent devices add complexity to the already complex implementation task for a man-machine interface (41:41). An inadequate implementation of concurrent devices can result in an error-prone and clumsy man-machine interface (5:199).

Prototyping. Prototyping software systems is a common practice. But prototyping a graphical man-machine interface has special problems. Prototyping shows the user how a system will operate and identifies problems before the system is actually coded. To do this, a prototype simulates the man-machine interface portion of the program. But to simulate a graphical man-machine interface almost requires the complete man-machine interface, which defeats the purpose of the prototype. Many present man-machine interfaces are only crudely prototyped on paper (26:249).

Many researchers are working on automated methods to prototype graphical man-machine interfaces, but their results are disputed (11:179-206; 25:51-59; 35:259-266).

Graphical Interface Applications and Examples

Overview. Graphical man-machine interfaces are used with a variety of application programs. Some of the interfaces are effective, others are not. To illustrate the characteristics and features of effective interfaces, five examples of very successful graphical man-machine interfaces are described here. The five examples are in the areas of; data-base maintenance, drafting and modeling, operating systems, system control, and teaching and training.

Data-base maintenance. Over a six year period, an engineering model of a man-machine interface for use by meteorologists to enter weather data was developed for a nation-wide data system. The system uses a high resolution raster color display, a menu-driven command structure, and a multi-color geographical map including weather contours. The screen consists of the map display with a menu area along the right side and bottom of the screen. Separate menu areas are used for the master menu, the submenus, and data entry menus. Three color schemes are used. The first color scheme, using white, green, and orange, is used for interactions with the user. The second color scheme uses

tan and brown to display the geographic map outline. The third color scheme, using 64 colors selected for high contrast from each other, is used for the weather contours on the map. The map control allows the operator to zoom to any selected area by selecting either a rectangular area or the center of the area desired. Many other weather-related functions are also interactively controlled by the operator. The system is a strong candidate for the National Weather Service operational workstations (23:147-156).

Drafting/modeling. Worldview is a computer-aided architectural and engineering design system which combines drafting and modeling approaches into a single CAD system. The graphical man-machine interface for the system is driven by a digitizing tablet or a mouse, is menu driven, and displays single or multiple two-dimensional and three-dimensional views of architectural designs. Multiple menus, both static and pop-up, flank the work area on three sides. The operator can choose between multiple design modes and can zoom and pan on any of the two-dimensional or three-dimensional displays. The program accepts both graphical and alphanumeric inputs. The system attempts to bridge the gap between modeling and drafting and is currently being used commercially and by college architectural students (15:36-46).

Operating systems. Some computer systems, like the Apple Macintosh, use graphical man-machine interfaces for

their operating system. The Macintosh's operating system is based on a previous Apple computer, called the Lisa, which is based on the Xerox Star computer. The Star computer was a result of research into graphics-based operating systems done at Xerox PARC (Palo Alto Research Center) (24:108; 42:30-31; 43:36). The Macintosh uses a mouse-window-desktop technology. The screen is arranged like a desktop, with icons representing computer functions. The interface uses dialog boxes, pull-down menus, multi-windows, and a mouse to create a data-as-concrete-object scenario. Apple designed the interface from the users point of view, and the result was a graphical man-machine interface which really is "user friendly" (42:31-54).

System control. NASA controls many communication and scientific spacecraft from ground control centers. The centers monitor screens of systems and sub-systems for a given spacecraft. Early NASA control systems used several CRT screens all displaying alphanumeric data. Each system needed several operators to monitor the CRTs. New NASA control systems use graphical displays and graphical man-machine interfaces to display the same amount of information to a single operator. One system, the Multi-Satellite Operations Control Center, uses several primary and secondary color screens, each with graphical system status displays and multiple menus. The system's displays use color graphics to represent models of the real world

systems. System control displays, using graphical primitives mixed with text, represent a system and it's operation. Color is used to represent status information. Each system status display is surrounded on three sides by a variety of menus allowing the operator to input commands to the system and to see the results of his inputs. The new graphical system has simplified the control process and reduced the cost and complexity of the control system (20:415-422).

Teaching/training. GUIDON-WATCH is a graphic interface used to give medical students access to a knowledge-based medical consultation system. The system is designed specifically to allow the student to view the reasoning process used by the knowledge-based system during medical consultation sessions. The interface uses a high resolution monochrome display, is mouse driven, and features multiple windows, pull-down menus, and an on-line help facility. Data is displayed to the users with tables and decision tree graphs using dynamic screen updates. By providing multiple views of information, GUIDON-WATCH helps the user to better understand the complex knowledge-based system (33:51-63).

Robotic Application Example

A graphical man-machine interface cannot be developed totally independent of the application program The

interface must reflect the application and must meet the technical requirements imposed by the application program. For this research effort, one operational robotics research application program was used as a representative application. The program, called ROBSIM (Robotic Simulation), is described in this section. Specifically, the purpose, capabilities, and structure of ROBSIM are outlined in the following paragraphs. The material in this section came from the NASA ROBSIM report and the ROBSIM User's Guide (28:1; 29:A-1).

Purpose. The purpose of ROBSIM is to provide a range of computer capabilities for use in the computer simulation of robotic systems. The program is designed to assist in the design, verification, simulation, and study phases of robotic system design and development (29:A-1).

Capabilities. ROBSIM provides the capability to perform kinematics and dynamic analysis of user-defined rigid link manipulators. The kinematic analysis provides positions, velocities, and accelerations of all parts of the manipulator for a prescribed motion. The dynamic analysis includes requirements analysis which calculates system loads for specific motions, and response simulation which calculates the motion resulting from a prescribed set of driving torques or feedback control law. ROBSIM can simulate multiple arms with movable bases. The user can define the motion trajectories for both the manipulators

and the movable bases. Position and/or force calculations are done for the bases as well as for the joints and links of the arms.

ROBSIM can use CAD/CAM (computer aided design/computer aided manufacture) generated data, written according to the Initial Graphics Exchange Specification, to model system components. The component model can be viewed by the user on a graphics display. Graphical displays of data, manipulator motions, and simulations are also provided (28:1).

Structure. The ROBSIM program is divided into four sections. The first section, the preprocessor function, handles the input, conversion, and display of CAD/CAM data. The second section is the system definition function. The system definition function is the part of the program used to create or modify simple or detailed arms, environments, targets, loads, or robotic systems. The third section, the analysis tools function, handles the response or requirements simulation. The fourth section of the program is the postprocessor function. The postprocessor function runs playbacks of the hardware motions or simulations and handles data outputs, both graphics and hardcopy (29:A-1).

Summary

Graphical man-machine interfaces combine diverse technical areas into a single software product. One of the

technical areas, graphical dialog techniques, is fairly well understood and many successful working examples exist. Software design and implementation research and practices are applicable to graphical man-machine interfaces and provide some guidance for interface design. But interface to user interaction, although heavily researched, is not yet well understood. Also, standard methods for combining these areas into a successful man-machine interface are not available (38:190-193; 40:238). At this time, successful interfaces seem to be more a result of skilled artisans and hard work than of "cookbook" approaches (11:205).

III. System Theory and Design

Overview

The knowledge outlined in Chapter II was used to create an effective and highly flexible interface design. The complete design created during the thesis research effort is described in this chapter. The first part of the design phase included defining the various requirements for the graphical man-machine interface. A design philosophy was formulated and a design approach developed before the design was created. The design itself consisted of two major parts, the interface displays and the interface software. The interface design included a variety of special features, which are also described in this chapter.

System Requirements

The requirements for this thesis effort vary widely. Many requirements are common to all man-machine interfaces, some are specific to color graphics man-machine interfaces, and a few are driven specifically by the application programs and the expected use of the system. In this section the requirements are divided into two parts, general user requirements and specific interface requirements.

General User Requirements. Four general user requirements must be met by this graphical man-machine interface. The four requirements are; error handling, variable user skill levels, user preferences on graphical displays, and on-line help.

Error handling. Many of the robotics software resources are hard to learn and allow users to easily make errors. One of the goals of this thesis effort is to improve on that aspect of the application programs. Therefore, the graphical man-machine interface should be designed to reduce or, where possible, eliminate the opportunity for the user to make errors.

Variable skill levels. The robotics researchers and students who will use the integrated evaluation environment vary widely in their experience with both the various application programs and with graphical man-machine interfaces. For this reason, the graphical man-machine interface should be configurable to meet the various needs and preferences of the users. The interface should include configuration options which allow the user to customize the interface, resulting in a comfortable and efficient man-machine interaction.

Graphical display preferences. Users have a wide variety of preferences concerning color graphical man-machine interfaces. Even though some of these preferences conflict, three of them are general requirements of this

thesis effort. The three requirements are as follows:

1. Provide fast system response, uncluttered screens, and a limited use of color.

2. Provide simple graphics, clear and simple controls, with the colors used for selections to contrast highly with the background and the other interface features.

3. Provide a system display which is aesthetically pleasing, with medium contrast and attractive color schemes.

On-line help. A goal of this thesis effort was to make the graphical man-machine interface easy to learn. New users should be able to use the interface without the need of training or reference manuals. To meet that goal, the interface should provide at least three different types of on-line help. The types of help are:

1. Help on the commands and options used by the interface and the application programs.

2. Help on the organization and structure of the interface and the application programs and how to navigate through them.

3. Help on how to use the various help functions.

Specific Interface Requirements. The specific requirements for this graphical man-machine interface came from the robotics application programs, from the integrated evaluation environment concept, and from the general user requirements discussed in the previous paragraphs. These specific requirements are divided into four areas. The areas are; integrated evaluation environment requirements,

application program requirements, hardware requirements, and software requirements.

Integrated evaluation environment requirements. The envisioned final integrated evaluation environment will include several robotics software resources. Therefore, the graphical man-machine interface must be designed to interface with multiple application programs. To support even further software additions, the interface design must be expandable. The general design should maintain consistency throughout the environment, with all screens and features remaining essentially the same for all of the application programs.

Robotics application requirements. The robotics application programs added two requirements that are not found among simple application programs. The first requirement is a multi-option selection capability that can handle more than 20 selections. The other requirement added by the robotics application programs is the need for the interface to clear its display from the computer screen to allow for graphical data displays. This also means the interface must be able to return its display to the screen when an application program is through with its graphics output.

Hardware requirements. The various requirements from the previous sections put high demands on the system hardware. To meet the demands, the hardware system must

have at least the following attributes:

1. A high resolution color monitor.
2. Both keyboard and mouse input devices.
3. The system must be fast, to perform the interface functions quickly and to not reduce the productivity of the users.

Software requirements. As with the system hardware, the previous requirements lead to specific software requirements for the graphical man-machine interface. The software requirements are:

1. The system must operate in a multi-program environment.
2. The system must operate in a multi-user environment.
3. The interface must use the VMS operating system.
5. The software code must be highly modular.
6. The software must be well documented.

Design Theory

A basic design theory was developed before the design of the graphical man-machine interface was begun. The theory consisted of two parts, a design philosophy and a design approach, both of which are described in the next two sections.

Design philosophy. The general philosophy was developed from the research completed for chapter two, from AFIT course work, and from personal experience. The philosophy consists of nine parts, which are as follows:

1. The graphical man-machine interface will not attempt to be "all things to all people".

2. The design should be minimally influenced by past personal experience, and should be heavily influenced by modern software design practice and by good examples of existing graphical man-machine interfaces.

3. The interface should not expect or require the user to be a computer expert.

4. KISS (keep it simple). Leave out fancy but unnecessary features, especially if they complicate the user's job or increase the effort needed to create the interface prototype.

5. Use the available hardware and software resources to reduce the time and effort required to create the interface.

6. Keep the interface displays simple. The displays should be functional, not a place to show off the creators skill or lack thereof.

7. The interface will be designed from the system level, just using part of the ROBSIM control structure as an application example.

8. Color will only be used to meet the requirements of the interface. Displays will not be garish, information indicated by color will also be indicated by text, and standard meanings will be assigned to standard colors (such as green means go, red means stop).

9. The interface prototype will be designed as a stand alone program, not as an add-on to an existing program.

Design approach. As with the design philosophy, a specific design approach was developed before design began. The approach consists of four parts, as follows:

1. Design from the user's point of view.

2. Design the displays and the operation of the interface first, then design the software needed to make the interface work.

3. Design from the top-down.

4. Start the design with the easiest of the interface functions.

Design Overview

The graphical man-machine interface prototype design is divided into two separate parts. The first part is the design of the user's view of the interface, or the screen displays. The second part is the design of the software needed for the interface. General descriptions of the two parts of the design are in the following two sections.

Interface Displays. The man-machine interface uses three types of displays to interface with the user. The types of displays are decisions screens, help screens, and configuration screens.

Decision screens. Decision screens are used to display to the user the interface and application options and to get selection input from the user. Besides information relating to user input functions, decision screens contain basic help, navigation, and system status information. Decision screens are the primary interface between the user and the application programs.

Help screens. The interface design uses two kinds of help screens, one type to display basic help text and one type to display structure information. Basic help screens contain advanced help information on user options, application and interface functions, and interface

operation. Structure screens contain graphical depictions of the interface and application program structures. The main purpose of structure screens is to aid the user in learning and in system navigation.

Interface software. The interface software design is based on the Seeheim interface program structure described in the design structure section of Chapter II (8:205-206). A complete design for an operational program consists of three components, the dialog control component, the presentation component, and the application interface component. The three components and their relationship to each other are shown in Figure 1. This thesis effort was

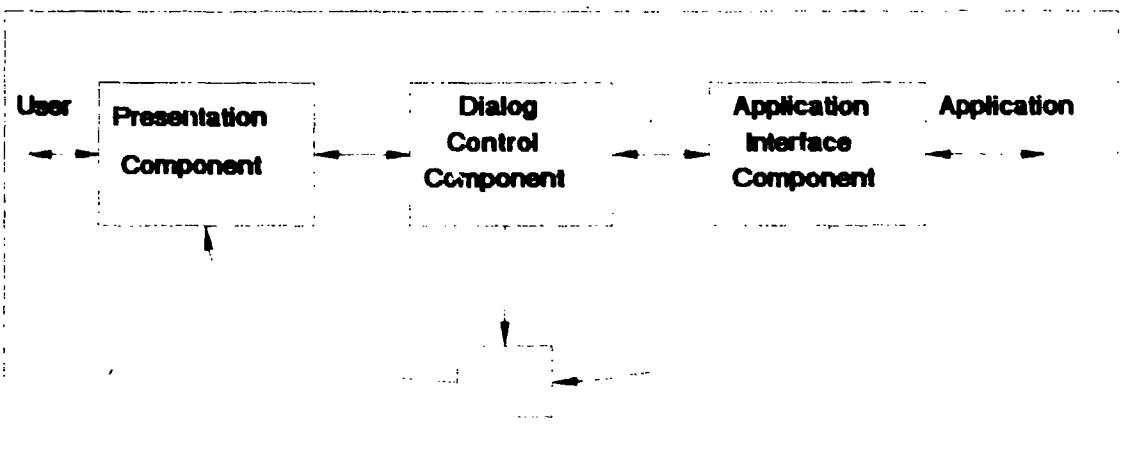


Figure 1. Seeheim Interface Program Structure (8:206)

only concerned with the design of the dialog control component and the presentation component.

Dialog control component. The dialog control component is the heart of the graphical man-machine interface. This component contains the program structure, control flow, and decision points. The dialog control component is independent of the graphics and interface drivers, and calls routines from the presentation component and the application interface component to handle those functions.

The structure and the control flow of the dialog control component are determined by, and must follow, the structure and control flow of the application program. For this thesis effort, part of the high-level ROBSIM program structure and control flow was determined using recursive transition networks. The dialog control component was then designed, also using recursive transition networks, to follow the partial ROBSIM structure and control flow.

Presentation component. The presentation component handles all of the screen displays and the I/O to and from the user. The presentation component is responsible for screen management, information display, input devices, interaction techniques, and lexical feedback. The presentation component functions are called by the dialog control component and it's design and implementation are device dependent.

Interface Displays

The interface displays are the only part of the graphical man-machine interface the user ever sees. Therefore, the user considerations outlined in Chapter II heavily influenced the display designs.

All of the interface displays use the entire computer screen. For design purposes, the screen was divided into 10 areas. The 10 areas are shown in Figure 2. Some

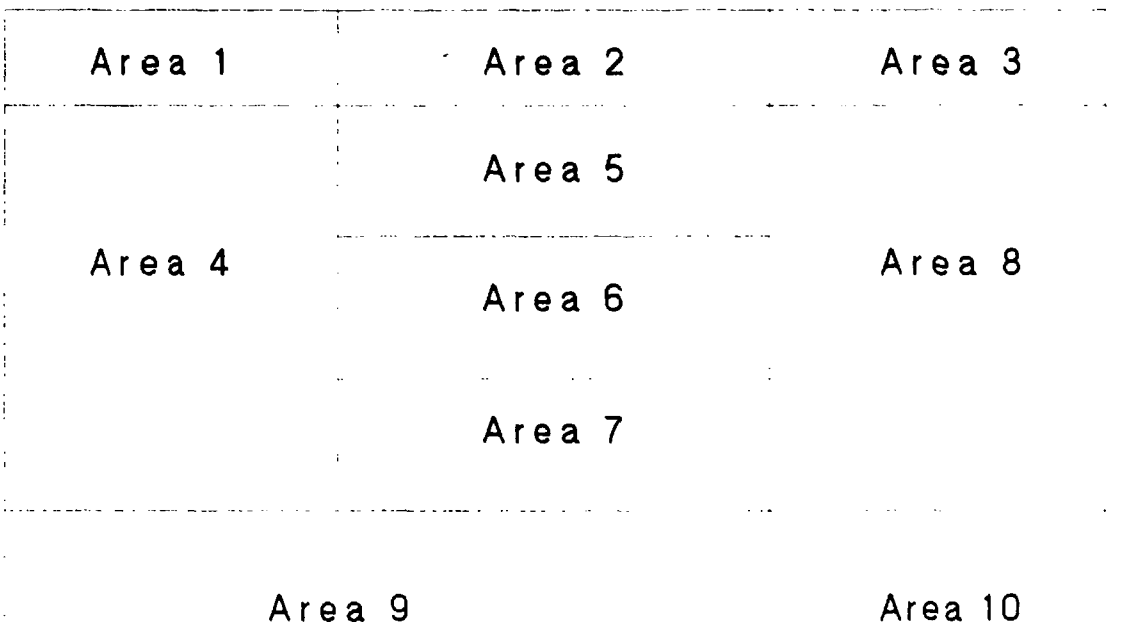


Figure 2. 10 Area Display Design

features are common among all of the displays. The basic color scheme of the interface uses shades of light blue, to

reduce contrast and to make the interface aesthetically pleasing. All of the text is displayed in black, and most of the areas are outlined in medium blue. The interface and the application programs are color coded, as shown below.

Interface:	White
Application 1:	Blue
Application 2:	Green
Application 3:	Red
Application 4:	Yellow

The various kinds of screens designed for the graphical man-machine interface are described in the following sections.

Decision screens. Figure 3 is a black and white example decision screen with all of the interface configuration options enabled (the configuration feature of the interface will be described later). The following paragraphs describe the screen areas used by the decision screen design. Area 10 is described first, since the entire operation of the interface revolves around the main menu.

Area 10, twelve option static menu. The purpose of the menu in area 10 is primary choice selection. The menu supports from one to twelve selection options, and is the menu used by the user to select paths through the interface program control structure. The basic menu color is a medium blue, with active menu option buttons displayed in green and labeled with the option labels. Inactive menu

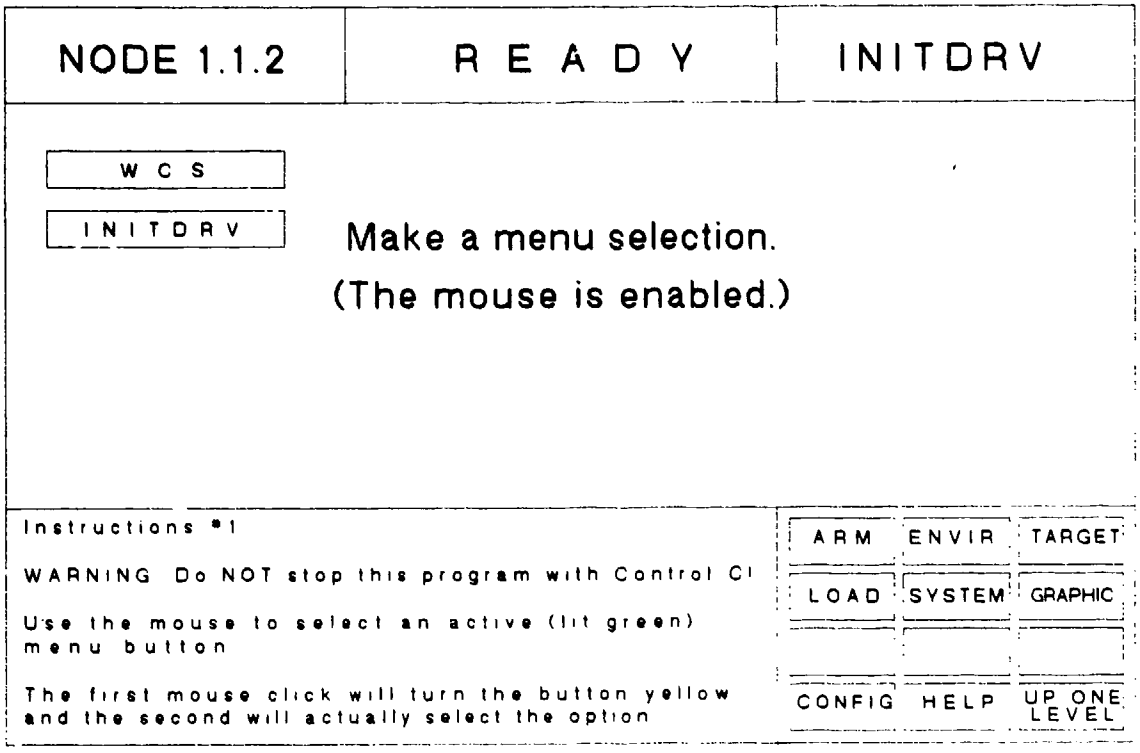


Figure 3. Example Decision Screen

buttons are blue. When the mouse button is pressed on an active menu button, the selected button turns yellow. When a button is selected twice in a row, the button turns red and the interface will not accept another input until the selected option has executed. When execution is complete, the main menu configures for the options at the new decision point, again turning active option buttons green and inactive buttons blue.

Area 1, screen identification. The purpose of area 1 is to display the node identification to the user. The control node number (described in the software design

section) is displayed in black text and the screen area is filled with the color code for the application program. The user has the option (described in the configure section) to disable the color option for area 1, which will result in the use of light blue as the area 1 fill color.

Area 2, system status. The purpose of area 2 is to display the system status to the user. When the interface is waiting for an input, this area is green and is labeled "Ready". When a menu button is selected once, area 2 turns yellow and is labeled "Option Selected". When a menu button is selected twice in a row, area 2 turns red and is labeled "Executing". The color option for area 2, like for area 1, can be disabled. If the color option is disabled, the interface uses light blue as the area 2 fill color. The text option can also be disabled for area 2, which will leave just the fill color in the area.

Area 3, application identification. The purpose of area 3 is to identify the application program, or the part of the program, that the user is presently in. If the area 3 color option is enabled, the fill color used in area 3 is the application color. And, as in areas 1 and 2, the interface uses light blue as the area fill color if the color option is disabled. If the text option is enabled, the name of the application program is displayed.

Area 4, (option A), command history. Area 4 of the screen normally displays the sequence of commands corresponding to the command path leading to the present decision point. The commands are displayed in descending order, each surrounded by a rectangle, matching in descending order the program levels traversed. Only commands leading from one decision node to another are displayed. The rectangles or the text can be disabled by the configuration option.

Area 4 (option B) 20 choice dynamic menu. A dynamic menu is only displayed in area 4 of the screen when more than 32 selection options are needed by the application program. The physical size of the menu is determined by the number of choices it must support, and can be configured to support any number of choices between 5 and 20. Each option is labeled with an option label and has a green area indicating the option is active. The indication areas change color with mouse selections in the same way as the active buttons in the main menu.

Area 5, interface prompt. The purpose of area 5 is to display the system prompt to the user. Prompts such as "Make a mouse selection." are displayed in this area.

Area 6, system response. Area 6 of the screen is reserved for system feedback to the user. The designed use is to tell the user when input data is incorrect. Data

input methods were not addressed by this thesis, so screen area 6 is reserved for future use.

Area 7, data input. The purpose of area 7 is to display data input back to the user. As with area 6, area 7 is reserved for future use.

Area 8, 20 choice dynamic menu. A dynamic menu is only displayed in area 8 when more than 12 selection options are needed by the application program. The dynamic menu used in area 8 operates exactly like the menu described for the B option of screen area 4.

Area 9, instructions. Area 9 displays simple instructions to the user concerning what the user needs to know at the present decision point. No information on the function of an option is given, unless it is needed for option selection. The instructions can be turned off with the configuration option.

Help screens. Screen areas 1, 2, 3, 9, and 10 work exactly the same in the help screens as they do in the decision screens. Therefore, only the differences between the help screen areas and the decision screen areas are described here.

Area 4-8, help text. The central part of the screen, consisting of areas 4, 5, 6, 7, and 8, is used by the help feature to display help text. The help text consists of descriptions of every decision option supported at the present decision point in the control

structure. The purpose of the help text is to provide new users with the interface and application information needed to learn the operation of the interface and the application.

Structure screens. Areas 1, 2, 3, 9, and 10 work exactly the same in the structure screens as they do in the decision and help screens. And as in the help screen case, only the differences between the decision screen areas and the structure screen areas are described here.

Area 4-8, structure. The central part of the screen is used by the interface structure feature to display graphical depictions of the control structure and control flow of the interface and the application programs. The graphics includes decision points, with node labels and command labels, and the structural relationship between them.

Interface Software

The interface software design is divided into two parts. Part one, the dialog control component, covers the design of the control component and the control flow used in this thesis effort. The second part of this section, the presentation component, covers the design of the various software components which make up the complete presentation component. All of the critical software components for both the dialog control component and the presentation component were designed using the pseudocode

shown in Appendix A. Both of the component designs are described in the following two sections.

Dialog control component. This part of the software design consists of two parts. Part one describes the control flow and structure used in the interface design. The second part is composed of individual descriptions of the modules which make up the complete dialog control component.

Control flow. The control flow of the dialog control component is based on a partial control flow structure taken from the ROBSIM application program. Only part of the ROBSIM control structure was needed to demonstrate the capabilities of the overall interface design. The ROBSIM control flow was determined by running the program and following the flow and options in the program. A part of the structure was determined from this effort, and that structure was represented using RTNs (recursive transition networks). RTNs were then used to implement the ROBSIM structure into the interface structure. The final design derived from this work is shown in the following 3 figures. Figure 4 shows the control flow for levels 0, 1, and 2 of the design, and Figure 5 shows the control flow for levels 2, and 3 of the design. Each ellipse represents a decision point, called a control node. The arrows indicate the paths in the structure that are open to the user. The structure shown in the two figures

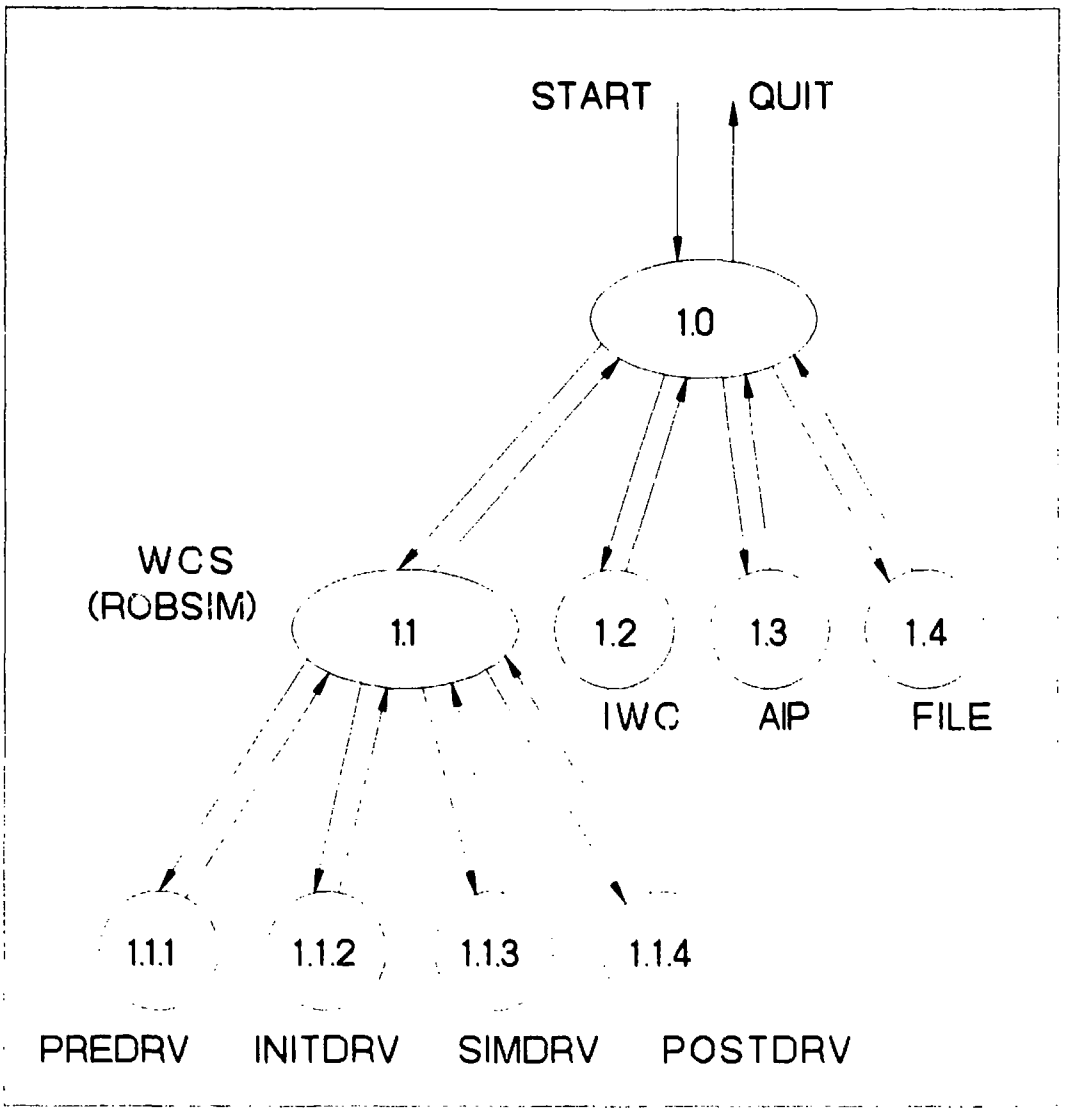


Figure 4. Levels 0, 1, and 2 Control Structure

represents the entire high-level control flow used in this design effort. But the RTNs do not show the control flow internal to a node. To meet the design requirements, the user must have access to interface functions and features that are not represented by the overall program structure. The control flow and structure required to support these

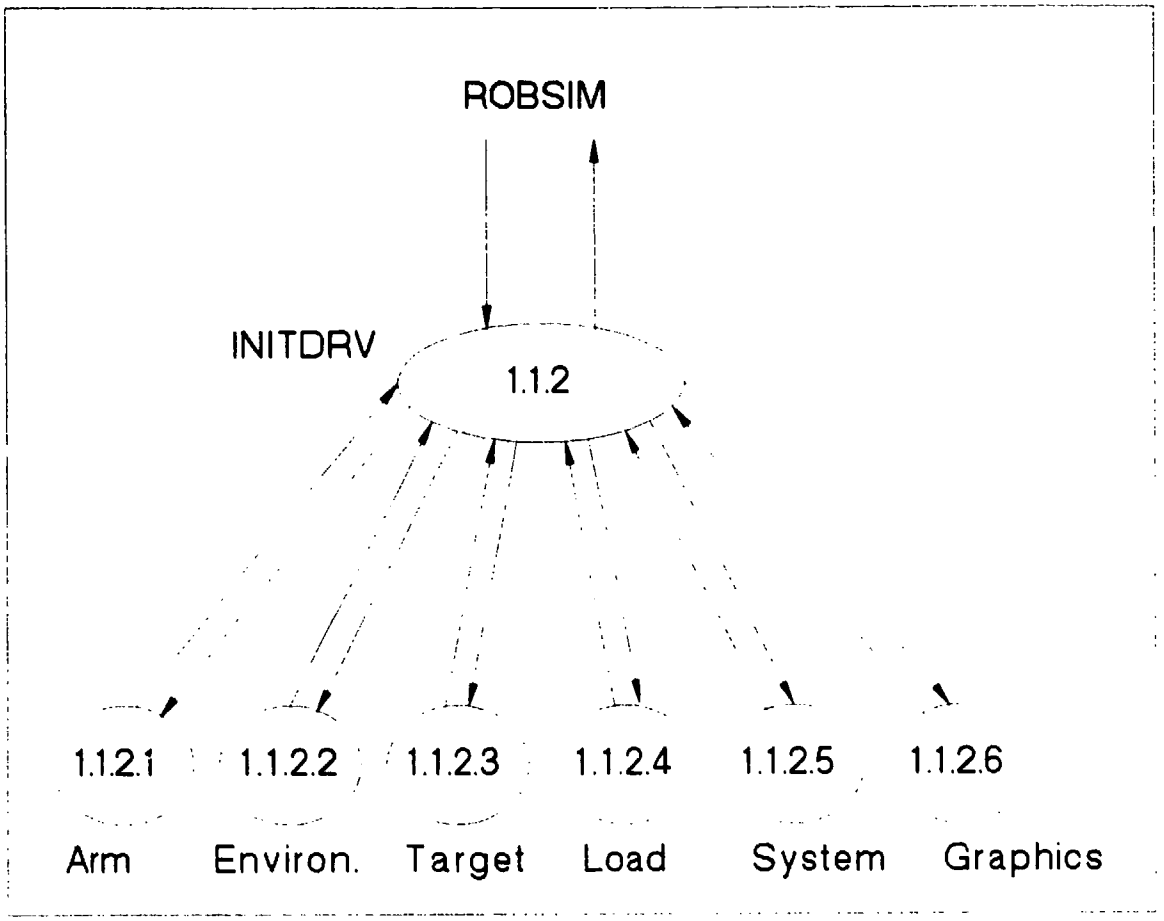


Figure 5. Levels 2 and 3 Control Structure

design features, for node 1.0, is shown in Figure 6. In this figure, interface options such as the help feature and the configuration feature are included in the internal control flow of node 1.0. This control flow also applies to node 1.1 and node 1.1.2. Combining Figures 4, 5, and 6 gives the complete design of the structure and control flow used for this thesis effort.

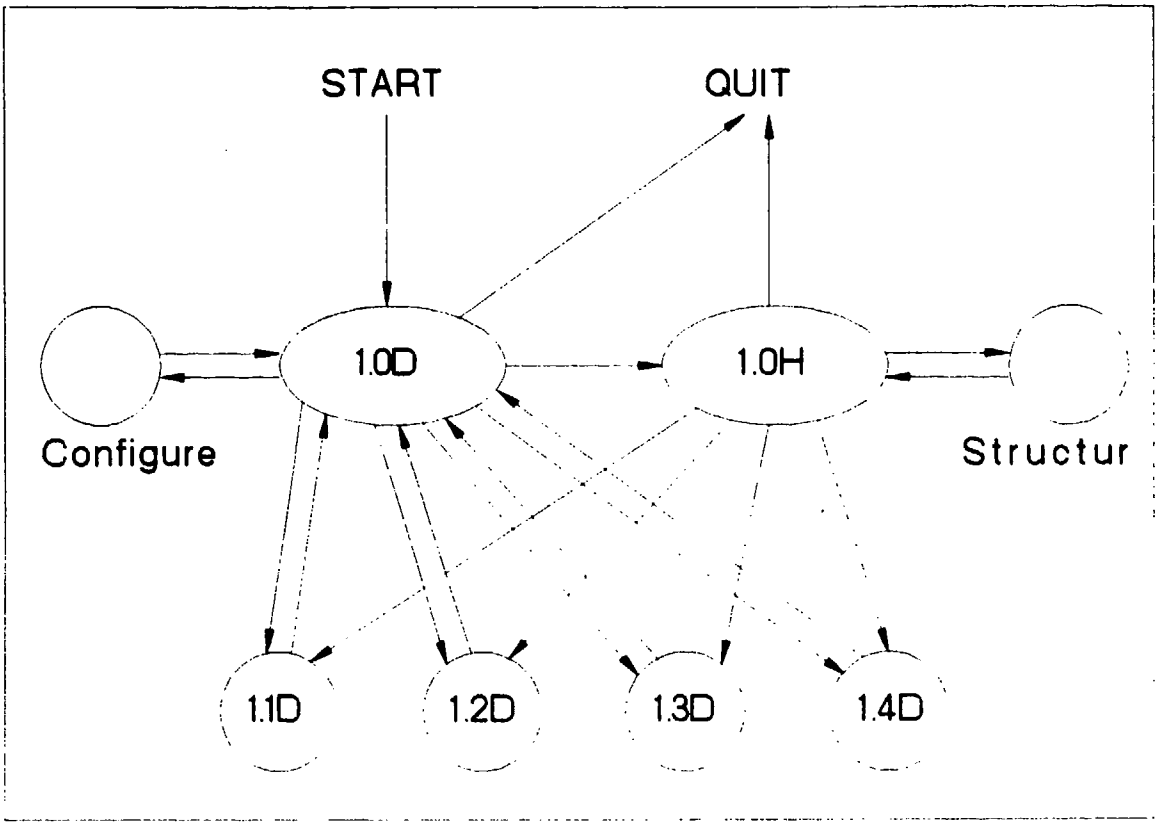


Figure 6. Node 1.0 Control Flow

Software modules. The dialog control component design actually consists of many individual dialog control module designs. Descriptions of the designs, which include decision nodes, help nodes, the configuration option, and the structure option, are in the following paragraphs.

Node 1.0D. Node 1.0D is the entry node into the interface and the software development environment. The node control structure allows the user to select between the four application programs, the help option, the interface configuration option, or the option

to quit the program. The four application programs are called WCS, IWC, AIP, and FILE.

Node 1.1D. Node 1.1D is entered from the 1.0 decision or 1.0 help module. The node is the entry point into the ROBSIM program structure and allows the user to select between four ROBSIM program options (nodes 1.1D to 1.4D), a help option, the configuration option, or the option to go back to node 1.0. The four ROBSIM program options are INITDRV, PREDRV, SIMDRV, and POSTDRV.

Nodes 1.2D, 1.3D and 1.4D. Nodes 1.2D, 1.3D, and 1.4D are called from the 1.0 decision or 1.0 help module. The nodes are the entry points into the IWC, AIP, and FILE programs, respectively. All three of these nodes are dead ends in this design. The only paths out of the modules are back to decision node 1.0.

Node 1.1.2D. Node 1.1.2D is entered from the 1.1 decision or 1.1 help module. The node is the entry point into the INITDRV part of the ROBSIM program structure and allows the user to select between a help option, to go back to node 1.1, the configuration option, or between six INITDRV program options (nodes 1.1.2.1D to 1.1.2.6D). The six INITDRV program options are ARM, ENVIRON, LOAD, TARGET, SYSTEM, and GRAPHICS.

Nodes 1.1.1D, 1.1.3D, and 1.1.4D. Nodes 1.1.1D, 1.1.3D, and 1.1.4D are called from the 1.1 decision or 1.1 help module. The nodes are the entry points into

the PREDRV, SIMDRV, and POSTDRV parts of the ROBSIM program, respectively. All of these nodes are dead ends in this design. The only paths out of these modules are back to decision node 1.1.

Nodes 1.1.2.1D to 1.1.2.6D. The six modules, node 1.1.2.1D to node 1.1.2.6D, are called from the 1.1.2 decision or 1.1.2 help module. The nodes are the entry points into six options of the INITDRV part of the control structure. All six of the nodes are dead ends in this design. The only paths out of the modules are back to decision node 1.1.2.

Help 1.0H, 1.1H, and 1.1.2H. The three help modules are entered from node 1.0D, 1.1D, and 1.1.2D, respectively. Each of the nodes is the help node for the respective decision node. The modules display the help text for the decision nodes and allow the user to select the interface structure option, the configuration option, the exit option, or any of the application options supported by the respective decision node.

Structure option. The structure module graphically displays the structure of the interface or application program relevant to the user's location in the program structure. The option is part of the help function and can only be accessed from a help node.

Configuration option. The configuration option is called from the decision or help nodes which

support the configuration option. The module allows the user to select from four interface configuration options. Option one is the full configuration option, which enables all of the interface screen area options, described in the custom configuration section. Option two is the basic configuration, which disables all of the interface screen area options. Option three, which calls the file configuration module, is a future option to allow the user to load a custom configuration from a disk file. The file configuration option is not included in this design. Option four, which calls the custom configuration module, allows the user to customize the interface configuration.

Custom configuration. The custom configuration module, which is called from the configuration module, allows the user to enable or disable individual interface options. The interface options are:

Option 1. Area 1 color on or off.

Option 2. Area 2 color on or off.

Option 3. Area 2 text on or off.

Option 4. Area 3 color on or off.

Option 5. Area 3 text on or off.

Option 6. Area 4 command history on or off.

Option 7. Area 9 instructions on or off.

File configuration. The interface design supports the future addition of an option to select an

interface configuration from a disk file, but the option itself is not included in this design.

Presentation component. The presentation component part of the design is divided into eight sections corresponding to the eight primary software functions that make up the presentation component. The eight sections are: begin graphics, set options, display screen, get user input, clear partial, clear to graphics, restore interface, and end of graphics.

Begin graphics. The begin graphics module sets all of the parameters and windowing functions needed for the interface graphics to function. Begin graphics also displays the basic interface decision screen on the computer screen.

Set options. The set options module sets the options used by the various software modules for the screen management and screen display functions. All of the options are represented by global variables, and are as follows:

> Screen option: The screen option variable indicates which screen type is used by a node. The three types of screen are decision screens, help screens, and structure screens. Using a global variable for this function allows additional types of screens to be added in the future.
>
> Level: The level variable indicates the level of the node in the program structure. The variable is used to display the correct level information on the command history option and for the structure display feature.

Screen Areas: The 7 screen area variables indicate which screen areas are active for a given node.

Static Menu: The 12 static menu variables indicate which selections are active on the static menu for a given node.

Dynamic 1: The dynamic 1 variable indicates whether the first dynamic menu is active, and if so, how many selections are active in the menu.

Dynamic 2: The dynamic 2 variable indicates whether the second dynamic menu is active, and if so, how many selections are active in the menu.

Static Label: The 12 static label variables contain the labels for the active selections on the static menu.

Dynamic 1 Labels: Same as above, except these 20 variables are for dynamic menu number 1.

Dynamic 2 labels: Same as above, except these 20 variables are for dynamic menu number 2.

Screen Label: The screen label variable contains the label used in area 1 of the screen.

Command: This group of variables contains the commands which were issued to reach the present node. The variables are used by the command history function and by the structure function.

Prompt option: The prompt option variable indicates what type of prompt is used by a node for area 5 of the screen.

System prompts: The system prompt variables contain the lines of prompt which are displayed to the user in area 5 of the screen.

Instruction Option: The instruction option variable indicates what type of instructions are used by a given node in area 9 of the screen.

Instructions: The instruction variables contain the lines of instructions which are displayed to the user in area 9 of the screen.

Get user input. The get user input module gets mouse selections from the user. The module uses the input options set by the set options module to determine if a mouse click is valid. If the click is valid, the module changes the selection button on the menu to yellow, changes area 2 to yellow, and waits for another mouse click. If the second input is valid and is the same selection as the first input, the module changes the selection button to red and changes area 2 to red. Get user input then returns the selection number to the calling module. If the second menu selection is not the same as the first, the module changes the first menu selection back to green, changes the second selection to yellow, and gets another mouse click. The cycle continues until two mouse clicks in a row are for the same menu selection. When two in a row are the same, the module changes the menu selection to red, changes area 2 to red, and gives the selection number to the calling module.

Display screen. The display screen module displays the interface screens using the options set by the set option module. Each of the nine screen areas is displayed, or not displayed, according to the screen option and the interface configuration options.

Clear partial. The clear partial module clears the interface screens for the display of a new screen. The

module clears only the screen areas that will change by the next call to the display screen module.

Clear to graphics. The clear to graphics module clears the entire interface display, except for button 12 on the main menu, from the computer screen. The purpose is to allow application programs to use the computer screen for their own displays.

Restore interface. The restore interface module redisplays the normal interface graphics after a call to the clear to graphics module.

End graphics. The end graphics software module cleans up the windows and graphics at the end of the run of the interface program, restoring the computer screen to the condition it was in when the interface program was started.

Special Design Features

To meet the wide variety of system requirements described earlier, the graphical man-machine interface design used a variety of features not ordinarily included in man-machine interfaces. The special features are described in the following paragraphs.

Feature 1. The interface uses a general purpose, configurable, 12 option mouse-driven main menu. The menu is always displayed, and is located in the lower right-hand corner of the screen to minimize mouse movement and to stay out of the center of the viewing area.

Feature 2. The interface screen design puts critical user information (prompts) in the center of the screen, right in front of the user. The user does not have to scan over the display to tell what the interface is doing.

Feature 3. Auxiliary user information is arranged around the center of the screen, out of the center of the viewing area. The information is available to the user if needed, but out of the way if not.

Feature 4. The design uses three types of special-purpose types of screen displays to meet the various user and application program requirements. Yet all three display types use one display design, the basic 10 screen area display design.

Feature 5. The 10 screen area basic display design supports a wide variety of configurations. Features can be easily modified or added to support present and future requirements.

Feature 6. The display of each of the 10 screen areas is designed to be independent of the display of every other screen area. Modifications to any one area can be made easily without affecting the other screen areas.

Feature 7. All of the control flow uses a generic control node design. Each node is modified slightly from the design to meet the individual node requirements, but every decision node is basically the same.

Feature 8. All of the screen information and configuration data that changes from one node to another is contained in one software module. The interface can be easily modified without changing the control flow or presentation software.

Feature 9. The entire interface configuration is easily modified by changing only a few global variables. The interface operation actually relies on a complete configuration change at every movement between control nodes.

Summary

As in the top-down approach to software design, the thesis design phase used an outside-in approach to man-machine interface design. What the expected user would need and want drove all of the screen designs and the operation concepts. Only after the user considerations were met did the design of the software begin.

The software was also designed from the view of the expected user. The design is simple, expandable, flexible, and modular. The software design will easily meet the demands of future robotics requirements. In addition, the simplicity and modularity of the design allowed for a straightforward and successful implementation process.

IV. System Implementatio

Overview

The complete implementation phase, from the implementation approach to the software modules, is described here. The design outlined in Chapter III was implemented the same way it was designed, in two parts. As in the design phase, part one is the dialog control component and part two is the presentation component. Both of the components, along with the general characteristics, the problems, and special features of the interface implementation, are described in the following sections.

Implementation Approach

As in the design theory section of Chapter III, a basic implementation approach was developed before the graphical man-machine interface design was implemented. The basic approach was based on six implementation precepts. The six precepts are:

1. Implement from the top down.
2. Test each function before continuing.
3. When possible, start the implementation from the most basic of the interface functions.
4. Implement the basic dialog control component functions first.

5. Implement the basic presentation components after the basic dialog control components.

6. Fill in the smaller functions only after all of the major functions are working.

Languages and Structure

The final implementation of the graphical man-machine interface used four major software components, a graphics software package, and two programming languages. The program organization and the implementation characteristics which used the various software resources are outlined in the following sections.

Languages and graphics packages. The implementation of the interface design needed to be fast to meet the design requirements. The C programming language is fast, is commonly used for graphics research, and was available on the GPX workstation. Therefore, the C language was picked for the complete interface design implementation.

The UIS system graphics and GKS were the graphics packages available on the GPX workstation used for the research. Both were capable of meeting the identified requirements for the interface. But the UIS package seemed faster than GKS, and UIS included functions which accessed many of the GPX screen management functions. Therefore, in the interests of speed and future expandability, the UIS graphics package was selected for the thesis project.

After part of the dialog control component was written and running in C, it was discovered that UIS graphics calls using character strings would not work from a C program on the hardware and software configuration used for the thesis effort. The UIS calls worked fine when called from code written in VAX FORTRAN. To keep the thesis effort on schedule, all of the subroutines using character strings were written in FORTRAN and called by C routines. Therefore, the final implementation is written mostly in C, uses some UIS functions directly, and uses other UIS functions through FORTRAN subroutines.

Program organization. The implementation of the interface is contained in four large software modules. The primary module, called int, which was written in C, contains the dialog control component. Another C module, called set2, contains the variable values used by the interface for all of the screen configurations throughout the program.

The presentation component is contained in two major software modules. The first, called pres and written in C, contains all of the presertation software modules that are called from the dialog control component. The second presentation component software module, grdriver, contains all of the UIS functions which would not work in the C language, and is written in FORTRAN. All of the grdriver modules are called by modules in the pres module. The code

for the four sections of software is contained in an AFIT Robotic Systems Laboratory Technical Report (16). Detailed descriptions of all of the software modules are in Appendix B.

Dialog Control Component

The dialog control component design described in Chapter III was successfully implemented during the implementation phase. The control structure, consisting of fifteen decision nodes, shown in Figures 4, 5, and 6 in Chapter III, was implemented almost exactly as designed. The only change made to the control structure was to allow the user to select the configuration option from either a decision or help node. Descriptions of the dialog control components, including decision nodes, help nodes, the configuration option, and the structure option, are in the following paragraphs.

Decision nodes. All of the decision nodes work basically the same way. Each node goes through five steps during execution. The first step is to set the variables which define the configuration used for the screen display corresponding to the executing node. When the configuration is set, the display screen module is called to put the configuration on the computer screen. Getting the mouse input from the user is the third step in the node execution. When the user has made a selection, the

node clears the configuration from the computer display and then executes the last step, step five. Step five is the only step that varies from node to node, and simply consists of executing whichever option the user selected during step three.

Help nodes. The help nodes operate using five steps just like the decision nodes. And steps one through four work exactly the same in the help nodes as in the decision nodes. Only step five, the final step, is different.

Help nodes are a selection option offered to the user from decision nodes. But help nodes also allow the user to select program options supported by the corresponding decision node. Step five supports this structure by doing one of two things. If the selected option is not a program option, step five executes the option. If the option is a program option, step five returns control to the fifth step of the corresponding decision node, and the decision node executes the program option.

Configuration option. The configuration option is implemented by using ten decision-like nodes in a linear structure. Each node allows the user to enable or disable one screen area option. The user steps through the entire sequence of ten questions, or can exit the configure option at any time by using a cancel option.

Structure option. The structure option consists of one decision-like node which supports five structure

screens. The five screens are described in the presentation component section. When the structure node is entered, the node determines which structure screen corresponds to the user's present location in the program structure, sets the options for that structure screen, and displays the structure screen to the user. The user has the option of viewing any of the five structure screens, or the user can exit the structure option.

Presentation Component

The major challenge of the implementation phase was the implementation of the presentation component. The complete implementation is described in two parts. The first part contains the general descriptions of the software components in the pres software module, and the second part contains the general descriptions of the software components in the grdriver module.

Pres module. The main functions of the presentation component, and all of the presentation modules called from the dialog control component, are in the software module called pres. Eight major routines are contained in pres, and overviews of their various functions are described in the following sections.

Begin graphics. The begin graphics module calls thirteen subroutines which prepare the system for the interface graphics and draw the permanent part of the

interface displays. Five of the subroutines called from begin graphics handle the overhead required to create and open the seven virtual displays used by the interface. Seven of the subroutines draw all of the permanent graphics, such as the menus and the structure screen graphics, onto the seven virtual displays. The last subroutine opens windows onto the virtual displays, putting the main display and the twelve option main menu on the computer screen. How the seven virtual displays are used by the interface is described in the following paragraphs.

Virtual display #1. The interface program uses the first virtual display for the main window and for the command history. The permanent main window graphics are drawn onto the virtual display during the execution of begin graphics.

Virtual display #2. The second virtual display is used by the interface program for the main twelve option main menu. The permanent parts of the menu are drawn during the execution of the begin graphics module.

Virtual display #3. All of the prompts used in area 5 of the screen are on virtual display number 3. The text for all of the prompts are written onto the virtual display during the execution of begin graphics.

Virtual display #4. The fourth virtual display is used for the instructions shown in screen area 9.

As with the prompts on virtual display #3, the text for all of the instructions is written onto the virtual display during the execution of begin graphics.

Virtual display #5. Virtual display #5 holds all of the help text for the basic help screens. Three complete help screens are drawn onto the virtual display, one for each of the three help node implemented in the interface program. As with virtual displays 3 and 4, all of the text is written onto the display at the beginning of program execution.

Virtual display #6. The sixth virtual display is used by the interface program for the structure option. Five complete structure screens are drawn onto the virtual display during the execution of begin graphics. The first screen contains the high level structure of the interface program and the first application program. The other four screens each contain the structure for one of the four application programs.

Virtual display #7. The last virtual display is used by the interface program for the two dynamic menus. The permanent parts of both of the dynamic menus are drawn onto virtual display #7 during the execution of the begin graphics module.

Set options. The set options subroutine sets the screen options for the screens used by the interface. Set options is called by the dialog control component before

every call to the display screen module. The module sets values for the global variables used by all of the presentation modules for screen management and configuration control. The design called for all of the labels used by the different display routines to be contained in the set options module. But the problem with the character strings in C required another structure. If the interface had been implemented as designed, the resulting program would have been more efficient, and all of the labels and options which change from screen to screen would have been confined to one subroutine instead of scattered throughout the program.

Display screen. The display screen module is called after the set options module at every control flow decision point in the interface. The display screen subroutine calls, or does not call, the various display area subroutines according to the screen type and screen area options. Each of the screen areas is displayed by its own separate software module. Four different methods are used by the various screen area display modules.

Method #1. Areas 1, 2, 3, and 10 are displayed by the most straight forward of the display methods. The respective display modules just redraw the colors and put new labels in the areas on every call to the display modules.

Method #2. Areas 5 and 9 do not draw or label anything when displayed. All of the text for the two areas is already written onto two virtual displays, as described in the begin graphics section. To display some of the text, the area 5 and 9 display modules just open a window onto part of the respective virtual display, and put the window into the proper location on the computer screen.

Method #3. The modules which display the two dynamic menus in areas 4 and 8 use a combination of the first two methods. The menus are permanently drawn on a virtual display, and windows are opened onto the virtual display as before. But the labels are drawn onto the menus just before the window is opened for each specific menu.

Method #4. Method number four is also a combination of the first two methods. The module which puts a command history in area 4 opens a window onto a virtual display and puts the window into the proper area. But the module draws all of the rectangles and writes all of the text to the virtual display just before the window is opened. In this way, the module can easily adapt to the various configuration options.

Get user input. The get user input module is responsible for getting mouse inputs, checking them for validity, and then passing valid mouse inputs to the calling dialog control node. The get user input module,

specifically the operation and the control flow, were implemented exactly as described in Chapter III, and the descriptions are not repeated here.

Clear partial. The clear partial software module determines what type of screen was last displayed and which screen areas need to be cleared before the next call of the display screen module. After determining which areas need to be cleared, the clear partial module calls subroutines to clear the selected areas.

Clear to graphics. The clear to graphics software module clears most of the interface from the computer screen so that application programs are free to use the screen. Clear to graphics closes the window to the main display and the window to the main menu, eliminating all of the interface display. The routine then resets the size of the main menu and reopens the main menu window, displaying only one main menu button in the lower right corner of the screen.

Restore interface. The restore interface software module closes the small window containing the one main menu button displayed by the clear to graphics module, then reopens the original windows to both the main display and the main menu, restoring the complete interface display.

End graphics. The end graphics module first closes the windows to the main display and the main menu.

End graphics then closes all seven of the virtual displays used by the graphical interface.

Grdriver. Many of the UIS functions of the presentation component, and all of the functions which use character strings, are in the software module called grdriver. The various software modules all perform one of four functions.

Function #1. Grdriver is responsible for putting the text and labels on all of the permanent graphics used by the interface. Specifically, the prompts, the instructions, the help text, and the structure screen labels are handled by software modules in grdriver.

Function #2. All of the nonpermanent labels used by the interface are also handled by grdriver. Every module in pres that displays a screen area which uses nonpermanent labels calls a software module in grdriver to put the labels on the screen area.

Function #3. The call to the UIS graphics package to open a window uses character strings. Therefore, a generic open window routine is in the grdriver module.

Function #4. The UIS asynchrounous trap routines which are used for mouse inputs also use character strings. So the routines which are used by the get user input module to get mouse inputs are in the grdriver software module.

Problems Encountered

Two problems were encountered during the implementation of the interface. The two problems, as well as the recommended solutions, are described in the following paragraphs.

Problem #1. The UIS graphics calls that use character strings in the passed parameters will not run on the GPX workstation when used by the C programming language. Because of the problem, the interface program is written in two programming languages. The interface could not be implemented exactly as designed and the resulting implementation is slower and harder to modify than intended. To fix the problem, the entire interface should be written, as designed, in one language. Either the problem with the GPX workstation should be fixed and the interface redone in C, or the interface should be redone in FORTRAN.

Problem #2. The VMS operating system allows the user to abort a program with the "Control C" command. But if the command is used to abort the interface program, the graphics terminal will lock up and the GPX workstation must be rebooted. The "Control C" command should be disabled by VMS options before the interface program is started, or the interface program itself should disable the command.

Special Implementation Features

The implementation of the graphical man-machine interface combined a variety of techniques in ways that are unusual in conventional computer graphics. The implementation made extensive use of multiple virtual displays, multiple windows, and various window management functions. Three unusual ways the interface used the techniques are described in this section.

Feature 1. The interface uses seven independent virtual displays. The use of multiple virtual displays allows the interface to use distinctive color schemes for the various interface screen features, and also allows the various features to be created and managed independently.

Feature 2. The interface uses multiple windows for every screen display. The multiple window implementation gives the interface the flexibility needed to support the various interface features and screen configurations.

Feature 3. The interface opens windows onto pre-created screen features and blocks of text as often as practical. The amount of graphical information which can be used is limited only by the available memory of the system. Also, real-time screen draws and text writes are kept to a minimum, which greatly simplifies the overall operation of the interface.

Summary

The interface implementation combines a wide variety of graphical techniques to provide a highly flexible graphical interface. The use of seven independent virtual displays creates an environment capable of very complex and effective color schemes. Multiple, independently managed screen areas gives levels of flexibility and adaptability greatly exceeding standard graphical man-machine interface implementations. The extensive use of pre-created graphics allows the addition of large amounts of future graphical information with no degradation of the operating speed of the interface. The effective combination of the various graphical techniques has resulted in a very powerful implementation. Also, the successful implementation of the interface allowed for the complete, operating interface prototype to be evaluated by a group of highly qualified users.

V. System Evaluation

Overview

A graphical man-machine interface can only be properly evaluated by a representative group of the final users. Therefore, for the final phase of this research, a group of AFIT students and staff evaluated the color graphical man-machine interface. The evaluation results were very good. The complete evaluation process, including the approach used, the evaluation itself, and the results, is described in the following sections.

Evaluation Approach

The graphical color man-machine interface prototype was evaluated by a group of AFIT faculty, students, and staff using a modified version of the CAD-Tool Human-Computer Interface Evaluation developed at AFIT (10:1-6). The Interface Evaluation was modified by the addition of a cover document to reflect the basic purpose of the interface prototype and to minimize the Interface Evaluation's emphasis on CAD tools. The CAD-Tool Human-Computer Interface Evaluation and the cover document are described later in this chapter. A copy of the Interface Evaluation and the cover document are in Appendix B. A

short description of the procedure used in the evaluation process is presented next.

Evaluation procedure. The evaluators were chosen as a representative group of typical AFIT users. The group had a variety of areas of expertise, with emphasis on robotics and graphics. The evaluators were each given a copy of the modified Interface Evaluation. They were not given any further instructions and were left to use the interface at a time of their own choosing. The evaluators were given the opportunity to remain anonymous if they desired.

Interface Evaluation Description

The interface evaluation consisted of a cover document, created specifically for this thesis effort, and the AFIT developed CAD-Tool Human-Computer Interface Evaluation.

Cover document. The cover document for the interface evaluation was created to describe the evaluation project, to outline the thesis effort, and to provide basic instructions to the evaluators. The cover document consisted of six parts, as follows:

1. An introduction to the cover document and the interface evaluation.
2. An evaluation overview and an outline of the general steps required to complete the interface evaluation.

3. A thesis description, covering the goals and scope of this thesis effort.

4. A summarization of the evaluation process.

5. A general description of the actual interface program and an outline of the basic program features.

6. All of the instructions needed for an evaluator to log on to a GPX, run the interface program, and complete the evaluation.

CAD-Tool Human-Computer Interface Evaluation. The Interface Evaluation was developed at AFIT to evaluate the interfaces used by AFIT CAD-Tool software resources. The Interface Evaluation contains questions concerning twelve aspects of an interface. The twelve aspects are:

1. System feedback.
2. Communication.
3. Error prevention.
4. Error recovery.
5. Documentation.
6. Expectations.
7. Confidence in the system.
8. Ease of learning.
9. Display of information.
10. Feeling of control.
11. Relevancy or system usefulness.
12. Overall evaluation of the system.

For the first eleven aspects, the evaluation uses six adjective pairs to describe each interface aspect. A

seven-interval range is used for the adjective pairs to indicate the evaluators feelings about each factor. Five of the adjective pairs refer to the interface aspect, and the sixth refers to the evaluators belief of the importance of the aspect. An example of how the evaluation uses the six adjective pairs is shown in Figure 7. Aspect twelve,

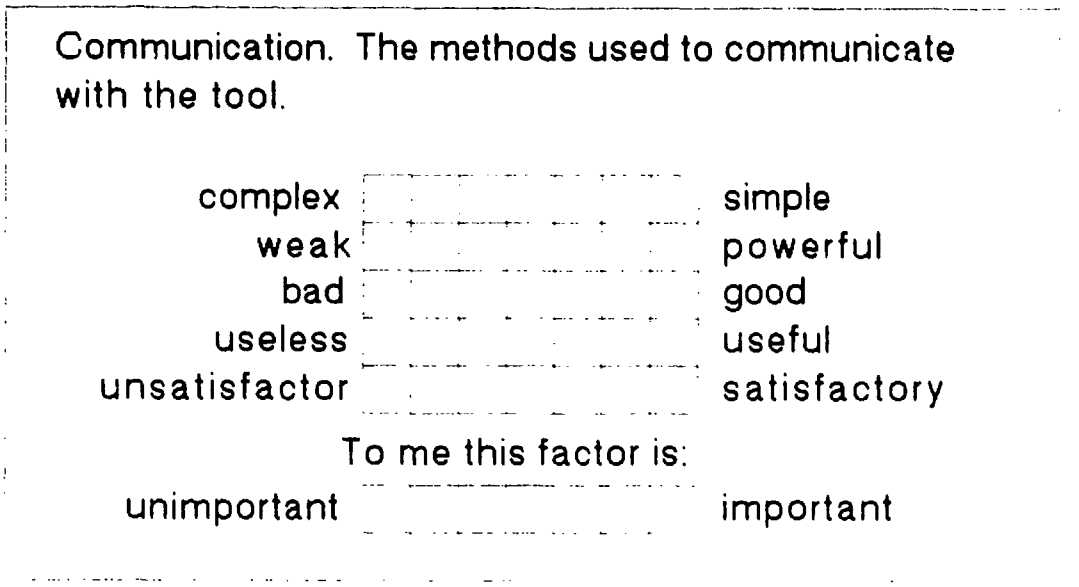


Figure 7. Example Adjective Pair Usage

overall evaluation of the system, uses a single adjective pair and seven-interval range. A more complete description of the evaluation operation is included in the Interface Evaluation's instructions.

Evaluation Results

Summaries of all of the areas directly relating to

the evaluation results are described in this section. The group of evaluators and the methods used to compile the evaluation results are described as well as an overview of the actual results for each of the twelve man-machine interface aspects used in the CAD-Tool Human-Computer Interface Evaluation. A summary of the comments, both specific and general, from the evaluations is also included. Detailed evaluation results are in Appendix D.

Evaluation overview. Brief descriptions of the relevant statistics of the group of evaluators and the methods used to compile the evaluation results are covered in the following paragraphs.

Evaluation statistics. Thirteen individual CAD-Tool Human-Computer Interface Evaluations were completed by AFIT students and staff during the interface evaluation. The evaluations were completed by three instructors, nine students, and by one member of the computer support staff. The breakdown of the evaluators, according to expertise, was three robotics people, three graphics people, and seven people which had neither robotics nor graphics experience.

The average time spent on the evaluations by the evaluators was 22.4 minutes. The evaluation period ranged from 15 to 40 minutes.

Results compilation method. Each of the adjective pairs used in the evaluation had a corresponding

seven-interval range, as shown in Figure 7. The range corresponded to negative levels of response on the left, a neutral response in the middle, and positive responses on the right. For the purposes of this evaluation, the seven response levels were assigned integer values ranging from negative three to plus three, as shown in Figure 8. Using this method, results around zero indicate neutral responses,

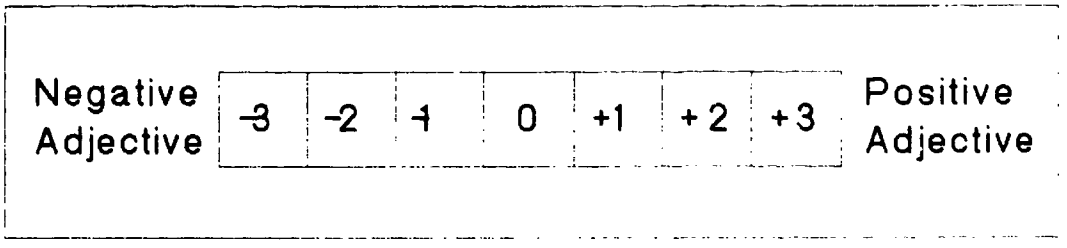


Figure 8. Integer Value Assignment

negative numbers indicate negative responses, and positive numbers indicate positive responses. For each evaluation, the responses for the five adjective pairs were averaged for each interface aspect. The averages for all twelve of the evaluations were then averaged, resulting in one number for each interface aspect.

Specific results. A summary of the results of the evaluation is shown in Table 1. The interface did very well in all areas, and did outstanding in many. Comments concerning important or unusual results are included in the following paragraphs.

Table 1. Evaluation Numerical Results

System Aspect	Importance	Result
1. System feedback	2.58	1.95
2. Communication	2.15	2.13
3. Error prevention	2.54	2.12
4. Error recovery	2.08	1.45
5. Documentation	2.09	.98
6. Expectations	1.17	1.39
7 . Confidence in the system	2.08	1.83
8. Ease of learning	2.54	2.56
9. Display of information	2.33	2.27
10. Feeling of control	1.82	2.67
11. Relevancy or system usefulness	1.82	1.58
12. Overall evaluation of the system		2.00

Aspect 1, system feedback. A goal of the thesis design was to let the user know what was happening at all times. System feedback, therefore, is an important aspect of the interface prototype. In addition, the evaluators felt that system feedback is an important aspect of man-machine interfaces in general. The evaluation results indicate the interface prototype was a solid success in this area. The performance ratings for this aspect were high and the comments were also positive.

Aspect 4, error recovery. The error recovery aspect of the prototype was not easily evaluated, and got widely varying results and a mixture of comments. Many evaluators felt the system handled basic errors well, but could not really evaluate the aspect accurately because the interface did not actually "interface" to a working

application program. In spite of the wide variety of responses, the performance ratings were still good.

Aspect 5, documentation. As with error recovery, the evaluators were not sure what to evaluate. The interface did not include any written documentation, as such, but did include on screen help text. The lack of paper documentation was intentional, since the prototype was designed to support new users.

The evaluators either ignored this aspect, gave it high marks for the available documentation, or gave it fair marks for lacking documentation.

Aspect 6, expectations. The evaluators were given no idea of what to expect, except from the evaluation instructions, and the comments reflected this. The evaluators also indicated they did not consider the expectation aspect to be very important. Since the thesis effort did not address expectations in any way, the comments and the performance ratings are also not very important.

Aspect 7, confidence in the system. The evaluators rated this aspect fairly well, and they also considered the aspect important. Confidence in the system was not directly addressed by the research effort, but the aspect is a direct result of other important factors which were addressed. As such, the performance ratings were important and very good.

Aspect 8, ease of learning. This aspect of graphical man-machine interfaces was directly addressed by the thesis effort and was a major goal. The evaluators were not given any help or operating instructions, other than from the interface itself, in order to specifically test this system aspect.

The evaluators felt the ease of learning aspect was important, and also felt the interface prototype was a solid success in this area. The comments were very positive and the performance ratings were excellent.

Aspect 9, display of information. The display of information aspect of the interface was an important thesis goal, and was also rated as important by the evaluators. And, as with aspect 8, the evaluators were very pleased with this aspect of the interface. All of the comments were good, as were the performance ratings.

Aspect 10, feeling of control. The feeling of control aspect of man-machine interfaces was not directly addressed by the research effort, but is a direct result of the success or failure of other important areas which were directly addressed. The evaluators also felt this aspect was important, but not as important as aspects 8 and 9. In any case, the evaluators gave the interface all positive comments and very high ratings.

Aspect 11, relevancy or system usefulness. As with aspect 5, documentation, the evaluators were not

always sure what to evaluate on this aspect. On the one hand, the interface did not actually "interface" to a working application program. On the other hand, the interface prototype had specific relevancy with respect to the thesis effort. And like the case of aspect 5, the evaluators gave varying comments or ignored the aspect. The evaluators did feel the aspect was important, and did give the interface good performance ratings.

General comments. Most of the general comments were very positive and related to how easy the system was to learn and use. Many evaluators liked the graphics and the use of a mouse for input.

On the negative side, the most common negative comment was about speed. A few evaluators thought the interface was too slow. Also, two evaluators disliked the mouse input, especially the two clicks required for a choice selection.

On the neutral level, a few evaluators did not have positive or negative feelings toward many aspects of the system, or the system itself. A few others felt the interface could not be accurately evaluated without a working application example.

Specific comments. Many of the specific comments, not concerning individual aspects, were suggestions on methods or features which may improve the system. One suggestion contained a good idea to streamline the configuration

option by using one menu, allowing the user to enable or disable all of the options from one decision point. Many of the ideas concerned features which were already incorporated into the system, but were apparently missed by the evaluator. On the negative side, a few of the evaluators found some of the help insufficient. Also, other comments indicated that evaluators had missed options or features entirely, which indicates either the evaluators did not take enough time with the system, or the help features were not giving the users enough information.

Discussion

The interface evaluation effort brought out many interesting points relating to the interface itself, the evaluation of man-machine interfaces, and man-machine interfaces in general. The relevant points are described in the following three sections.

Interface findings. Four minor points concerning the graphical man-machine interface prototype were brought out during the interface evaluation process.

1. People did not know the interface speed could be increased by disabling screen options, and the interface help features did not mention the speed considerations.

2. Some evaluators had trouble evaluating the interface when the system did not interface to a working application program.

3. Because of the limited control structure implementation, some evaluators could not

understand the purpose of a few interface features.

4. The evaluations brought out a few details which were overlooked during the design and implementation phase.

Evaluation findings. The evaluation process brought out three points concerning the evaluation of graphical man-machine interfaces.

1. Many people did not read the entire instruction sheet. From the comments made on the evaluations, many of the results may be incorrect.

2. A few of the evaluators did not test all of the functions. They made suggestions to add features already incorporated in the interface.

3. The evaluators could only evaluate what they saw, which was only a part of the thesis effort. The evaluation results do not really apply to large portions of the interface design or implementation.

General man-machine interface findings. Finally, two points concerning man-machine interfaces in general were made apparent during the evaluation process.

1. The evaluators made many recommendations concerning interface features and options which supported the information contained in Chapter II.

2. The personal preferences of the evaluators varied widely, again as suggested by the literature review.

Summary

The graphical man-machine interface was successfully evaluated by a representative group of end users. The process of having a variety of users evaluate a software system was very interesting and educational. Having the

users evaluate the system with no instructions, training, or help was also very risky. But the interface prototype performed very well during the evaluations and received outstanding ratings. The overall evaluation results were excellent, with a performance rating average of +1.91. The evaluation results indicate that the design and implementation phases of the thesis effort were a complete success.

VI. Conclusions and Recommendations

Summary of Results

An effective and efficient man-machine interaction is a smooth, flowing process usually requiring a high level of user proficiency. To be effective, especially for inexperienced users, a graphical man-machine interface must do most of the work of the man-machine interaction, and must also provide the user with a comfortable and productive environment.

To create the above conditions, a graphical interface must be a constantly changing, yet smoothly operating mechanism. The creation of such a mechanism is a highly complicated process. Many technical and nontechnical areas of knowledge must be correctly combined and finely balanced or the resulting interface is worse than useless.

The overall purpose of the thesis effort was to create such a smoothly operating mechanism for use with the various AFIT research software resources. During this thesis effort, the general user requirements and specific interface requirements were determined. A general purpose interface design was created to meet the wide variety of requirements. The design was implemented using combinations of graphical techniques to create a highly flexible interface. And, in the final phase of the

research, the interface prototype was evaluated with outstanding results.

The interface design supports four application programs, and can easily be expanded. The design met all of the general user requirements and specific interface requirements outlined in Chapter III. In addition, the design is flexible enough to be easily modified to meet future, as yet undetermined, requirements. The interface provides all of the expandability and flexibility needed to support the various AFIT software resources and the variety of expected users.

As with the design, the interface implementation met all of the system requirements and can also be easily expanded or modified. The entire implementation is based on flexibility, and allows the interface to be reconfigured to meet the present software resource requirements and the various user requirements, as well as any future demands.

The operating interface prototype was evaluated by thirteen highly-experienced users with very good results. All of the interface aspects addressed by the thesis effort, and many not addressed, received high ratings from the evaluators. In addition, the evaluation brought out a few good ideas for improvements or additions to the interface.

Conclusions

The color graphical man-machine prototype developed during this thesis effort will be used by researchers to drastically reduce learning periods and to increase the overall productivity of AFIT researchers and students. In addition to AFIT research, the prototype developed here can be easily adapted for use in any environment, government or civilian, which uses multiple, complex, application programs. The interface prototype features many excellent user support features, and would therefore be useful in environments which include users with a wide variety of skill levels.

The methodology used to create the man-machine interface resulted in a very successful interface on the first try. The methodology can be used by researchers or developers as an effective tool to successfully create other effective graphical man-machine interfaces.

Recommendations

Software enhancements. Even the best software projects can be improved upon, and the graphical man-machine interface created for this thesis effort is no exception. The following software enhancements would improve the program:

1. The user should be allowed to exit the program from anywhere in the interface.

2. The help facilities should be expanded. The users get a lot of information, but still not enough.

3. Expand the configuration feature. More options should be added, such as a left-handed display option, and options to change the interface color scheme.

4. Optimize the code. The interface needs to be faster.

5. Improve on the aesthetics of the interface. The quality of the text can be improved, and the colors adjusted slightly.

6. Add the capabilities to save custom configurations to disk files, and to load configurations from disk files.

Further research. A few areas of further research should be addressed. The areas concern completing the original purpose of the interface and adding to the effectiveness of the interface.

Various AFIT research software resources should be integrated into the interface. The final effectiveness of the interface cannot be tested, evaluated, and improved until the interface is operating with multiple working application programs.

The interface should provide the user with the capability to multi-task the application programs from inside the interface. To support this capability, the interface must allow the user to move around, within limits, in the integrated evaluation environment.

Advanced training methods, such as animated sequences, should be added to the interface. Animated

training sequences, for both the operation of the interface and the operation of the various application programs, would give new users all the help they need, and would greatly reduce training time.

Appendix A: Pseudocode Used In The Design Phase

Node 1.0, Decision Node

```
do_node1_0_D ()
    {
     answer1_0_D = go;

     while (answer1_0_D != quit)
         {
          set_options (101)
          display_screen
          get_user_input (answer1_0_D);

          if (answer1_0_D == help_option)
              {
               clear_partial (to help);
               do_help1_0_H;
              }

          if (answer1_0_D == 1_1 or 1_2 or 1_3 or 1_4)
              {
               clear_partial (to decision)

               if (answer1_0_D == node1_1)
                    do_node1_1_D

               else if (answer1_0_D == node1_2)
                    do_node1_2_D

              else if (answer1_0_D == node1_3)
                    do_node1_3_D

              else if (answer1_0_D == node1_4)
                    do_node1_4_D
              }
              }

          if (answer1_0_D == configure)
              {
               clear_partial (to configure)
               do_configure
              }
          clear_partial (to decision)
          }
```

Main Program

```
main()
    {
     start_up

     display_decision

     do_model_O_D

     shut_down
    }
```

Display Screen

```
display_screen
{
if (screen_area(1) == 1)
     display_area_1

if (screen_area(2) == 1 and area_option(2) > 0)
     display_area_2 (green)

if (screen_area(3) == 1 and area_option(3) > 0)
     display_area_3

if (screen_area(4) == 1 and area_option(4) > 0)
     display_area_4A

if (screen_area(4) == 2)
     display_area_4B

if (screen_option = decision or configure)
     display_area_5

if (screen_option = help)
     display_area_4-8H

if (screen_option = structure)
     display_area_4-8S

if (screen_area(8) == 1)
     display_area_8

if (screen_area(9) == 1 and area_option(9) = 1)
     display_area_9

if (screen_area(10) == 1)
     display_area_10
}
```

Set Options

```
set_options (screen_no)
int screen_no
{
if (screen_no == 101)
    {
     set_colors (application_1)

     screen_option = decision;
     application = operating system;
     level = zero;

     screen_areas()  = (1, 1, 1,
                        1, 1, 1,
                        1, 1, 1);

     static_menu() = (1, 1, 1,
                      1, 0, 0,
                      0, 0, 0,
                      1, 1, 1);

     no_of_choices = 4
          command(1) = "@robsim"
          command(2) = "   "

     dynamic_1 = 0;
     dynamic_2 = 0;
     static_label_1 = "ROBSIM";
     static_label_2 =
       etc.
     screen_label = "Decision Screen, Node 1.0";
     command_2 = "ROBSIM";
     system_prompt = "Select an Option";
     instructions = instructions 4;
    }

if (screen_so == 102)          etc. for all screens.
    {

    }
}
```

Get User Input

```
get_user_input (return choice)
{
pass = 0
return ready = no
```

```
while (return_ready = no)
{
valid input = no

while (valid input = no)
    {
     get_input (answer)
     test input (valid input,  yes or no)
    }

if (pass = 0)
    {
     pass = 1
     input_one = answer
     light_choice (answer, yellow)
     display_area_2 (yellow)
    }

else
    {
     input_two = answer

     if (input_one = input_two)
         {
          light_choice (answer, red)
          display_area_2 (red)
          return_ready = yes
         }
      else
         {
          light_choice (input_one, green)
          light_choice (input_two, yellow)
          input_one = input_two
         }
      }
  }

return answer

}
```

Generic Decision Node, No Option Choice From Help

```
do_node1_0_D ()
{
answer1_0_D = go;

while (answer1_0_D != stop)
    {
     display_screen (101);
```

```
        get_user_input (101, answer1_0_D);

        if (answer1_0_D == 1_1 or 1_2 or 1_3 or 1_4)
            (
             clear_D_screen_to_D

             if (answer1_0_D == node1_1)
                  do_node1_1_D

             else if (answer1_0_D == node1_2)
                  do_node1_2_D

             else if (answer1_0_D == node1_3)
                  do_node1_3_D

             else if (answer1_0_D == node1_4)
                  do_node1_4_D

             clear_D_screen_to_D
            )
           )

        if (answer1_0_D == help_option)
            (
             clear_D_screen_to_H;
             do_help1_0_H;
             clear_H_screen_to_D;
            )

        if (answer1_0_D == configure)
            (
             clear_D_to_C
             do_configure
             clear_C_to_D
            )
        )
```

Node 1.0, Help Node

```
     do_help1_0_H ()
     {
     answer1_0_H = go;

     while (answer1_0_H != quit)
         {
          set_optionws (102)
          display_screen
          get_user_input (answer1_0_H)

          if (answer1_0_H == structure)
              (
```

```
                clear_partial (to structure)
                do_structure (10)
                clear_partial (to help)
               )

          if (answer1_0_H == node1_1)
               (
                answer1_0_D = node1_1
                answer1_0_H = quit
               )

          if (answer1_0_H == node1_2)
               (
                answer1_0_D = node1_2
                answer1_0_H = quit
               )

          if (answer1_0_H == node1_3)
               (
                answer1_0_D = node1_3
                answer1_0_H = quit
               )

          if (answer1_0_H == node1_4)
               (
                answer1_0_D = node1_4
                answer1_0_H = quit
               )

          if (answer1_0_H == quit)
                answer1_0_D = quit

          }
     }
```

Do Configure

```
     do_configure ()
     (
     set_options (99)
     display_screen
     get_user_input (answer_con2)

     if (answer_con2 == full)
          (
           area_options = (1, 1, 1,
                           1, 1, 1,
                           1, 1, 1)
          )
     if (answer_con2 == basic)
          (
```

```
        area_options = (0, 0, 0,
                        0, 0, 0,
                        0, 0, 0)
    )
    if (answer_con == from file)
        do_file_config

    if (answer_con == custom)
        do_custom ()
)
```

Do Custom

```
do_custom ()
(
    set_options (991)
    display_screen                      ask if color
    get_user_input (answer_2)           for area 1

    if (answer_2 == no_color)
        area_option(0) = no_color   (0)

    if (answer_2 == color)
        area_option(0) = color      (1)

    set_options (992)
    display_screen                      ask if color
    get_user_input (answer_3)           for area 2

    if (answer_3 == no color)
        area_option(1) = no color    (0)

    if (answer_3 == color)
        area_option(1) == color       (1)

    set_options (993)
    display_screen                      ask if text
    get_user_input (answer_4)           for area 2

    if (answer_4 == no text)
        area_option(2) == no text    (0)

    if (answer_4 == text)
        area_option(2) == text       (1)

        etc.

    set_options (999)
    display_screen                          ask if save
    get_user_input (answer_disk)            config to disk
```

```
        if (answer_disk == yes)
            {
            save_config_to_disk
            }
        }
    }
```

Node 1.1, WCS (ROBSIM) Node

```
    do_node1_1_D ()
    {
    answer1_1_D = go

    while (answer1_1_D != back up one level)
        {
        set_options (111)
        display_screen
        get_user_input (answer1_1_D)

        if (answer1_1_D == help_option)
            {
            clear_partial (to help)
            do_help1_1_H
            }

        if (answer1_1_D == 1_1 or 1_2 or 1_3 or 1_4)
            {
            clear_partial (to decision)

            if (answer1_1_D == node1_1_1)
                do_node1_1_1_D

            else if (answer1_1_D == node1_1_2)
                do_node1_1_2_D

            else if (answer1_1_D == node1_1_3)
                do_node1_1_3_D

            else if (answer1_1_D == node1_1_4)
                do_node1_1_4_D
            }
            }

        if (answer1_1_D == configure)
            {
            clear_partial (to configure)
            do_configure
            }
        clear_partial (to decision)
        }
```

Node 1.2, IWC Node

```
do_node1_2_D ()
{
answer1_2_D = go;

while (answer1_2_D != go up one level)
     {
      set_options (121)
      display_screen
      get_user_input (answer1_2_D);
      }
}
```

Node 1.3, AIP Node

```
do_node1_3_D ()
{
answer1_3_D = go;

while (answer1_3_D != go up one level)
     {
      set_options (131)
      display_screen
      get_user_input (answer1_3_D);
      }
}
```

Node 1.4, FILE Node

```
do_node1_4_D ()
{
answer1_4_D = go;

while (answer1_4_D != go up one level)
     {
      set_options (141)
      display_screen
      get_user_input (answer1_4_D);
      }
}
```

Node 1.1, Help Node

```
do_node1_1_H ()
{
answer1_1_H = go;
while (answer1_1_H != quit)
     {
```

```
        set_options (112)
        display_screen
        get_user_input (answer1_1_H)

        if (answer1_1_H == structure)
            (
             clearH_to_S
             do_structure (11)
             clearS_to_H
            )

        if (answer1_1_H == node1_1_1)
             answer1_1_D = node1_1_1

        if (answer1_1_H == node1_1_2)
             answer1_1_D = node1_1_2

        if (answer1_1_H == node1_1_3)
             answer1_1_D = node1_1_3

        if (answer1_1_H == node1_1_4)
             answer1_1_D = node1_1_4

        if (answer1_1_H == up one level)
             answer1_1_D = up one level

    }
}
```

Display Area 1

```
display_area_1 ()
{
if (area_option(1) = color included)
     color = application_color
else
     color = background color

clear_area (x1, y1, size, color)
label_area (x1, y1, size, screen_label)
}
```

Display Area 2

```
display_area_2 (status color)
{
if (area_option(2) > 0)
    {
     if (area_option(2) = color on) (2 or 3)
```

```
            color = status color
        else
            color = backround color

        clear_area (x2, y2, size, color)

        if (area_option(2) = text) (1 or 3)
            label_area (x2, y2, size, area_2_label)
      }
}
```

Display Area 10

```
display_area_10 ()
{
if (static_menu (one) = yes)
    {
     clear_area (x, y, size, green)
     label_area (x, y, size, static_label_1)
    }
else
     clear_area (x, y, size, blue)

if (static_menu (two) = yes)
    {
     clear_area (x, y, size, green)
     label_area (x, y, size, static_label_2)
    }
else
     clear_area (            blue)

if (static_menu (three) = yes)
    {
     clear_area (          , green)
     label_area (         , static_label_3)
    }
else
     clear_area (           , blue)


etc. to static 12.

}
```

Do Depth

```
do_depth ()
{
if (level > 1)
    {
```

```
        clear_area (x, y, size, depth_color_2)
        if (level > 2)
            {
             clear_area (x, y, sixe, depth_color_3)
             if (level > 3)
            {
             clear_area (x, y, size, depth_color_4)
             if (level > 4)
            {
             clear_area (x, y, size, depth_color_5)
             if (level > 5)
            {
             clear_area (x, y, size, depth_color_6)
             if (level > 6)
                  clear_area (x, y, size, color_black)
            }
          }
        }
      }
    }
  }
```

Display Area 4a

```
display_area_4A
{
if (area_option (four) > 0 and level > 0)
    {
     if (area_option = no color)
          color = backround_color
     else
          color = application_color

     label_area (x, y, size, history_title)

     draw_outline (x1, y1, size, outline color)
     clear_area (x2, y2, size, color)
     label_area (x3, y3, size, command_1)

if (level > 1)
    {
     draw_outline (x4, y4, size, outline color)
     clear_area (x5, y5, size, color)
     label_area (x6, y6, size, command_1)

if (level > 2)
    {
     draw_outline (x7, y7, size, outline color)
     clear_area (x8, y8, size, color)
     label_area (x9, y9, size, command_1)
```

```
if (level > 3)

          etc. until will handle all needed levels

    }
  }
}
}
```

Display Area 5

```
display_area_5
{
if (prompt option = one)
     open window(x1, y1, size)

if (prompt option = two)
     label_area (x1, y2, size)

if (prompt option = three)

     Etc., Etc.
}
```

Display Area 9

```
display_area_9
{
if (instruction_option = one)
     open window(x1, y1, size)

same as code for display area 5

}
```

Light Choice

```
light_choice (choice, color)
{
if (choice = 1)
    {
     clear_area (x1, y1, size, color)
     label_area (x1, y1, size, static_label (1))
    }

if (choice = 2)
    {
     clear_area (x2, y2, size, color)
```

```
        label_area (x2, y2, size, static_label (2))
       }

  if (choice = 3)
       {
        clear_area (x3, y3, size, color)
        label_area (x3, y3, size, static_label (3))
       }

  same for all supported choices

  }
```

Display Area 3

```
  display_area_3 ()
  {
  if (area_option(3) > 0)
       {
        if (area_option(3) = color)
             color = application color
        else
             color = backround color

        clear_area (x3, y3, size, color)

        if (area_option(3) = text)
             label_area (x3, y3, size, area_3_label)

        if (area_option(3) = depth)
             do_depth
       }
  }
```

Appendix B: Detailed Software Descriptions

Overview

This appendix contains two sections. The first section describes the implementation of the dialog control component part of the graphical interface design, and the second section, the presentation component, describes the implementation of the software modules that make up the presentation component.

Dialog Control Component

The basic building block of the dialog control component is the control node, as described in the software design section of Chapter III. This section describes the implementation of each of the nodes completed in this thesis. The section is divided into four parts. Part one describes the implementation of the decision nodes. Part two describes the implementation of the three help nodes completed for this effort. Part three describes the implementation of the interface configuration feature, and part four describes the implementation of the interface's structure feature. To make the node descriptions complete, all of the descriptions include the labels and application colors used by the nodes.

Decision nodes. Four program levels, consisting of fifteen decision nodes, were implemented for this thesis. Only three of the decision nodes work like a decision point in an operational program. The three nodes are described in detail. The other twelve nodes are dead end nodes which were used to complete the partial control structure and to demonstrate various features supported by the interface design. The descriptions of the implementation of the twelve dead end nodes include descriptions of the features demonstrated and the node configurations. Node 1.1.1D was used to demonstrate the capability of the interface to clear the interface graphics from the computer screen, as described in Chapter III.

Node 1.0D. Node 1.0D is configured as the entry node into the interface. The node supports four application choices, a help option, the configuration option, and the option to exit the program. The display configuration is a normal decision node configuration with

areas 1, 2, 3, 5, and 9 active. Area 1 is labeled "NODE 1.0", area 3 is labeled "GMMI", area 5 contains the "Make a mouse selection." prompt, and area 9 contains basic instructions on how to use the mouse. The color code used in areas 1 and 3 is white.

Node 1.1D. Node 1.1D is configured as the entry node into the ROBSIM program structure. The node supports four ROBSIM choices, a help option, the configuration option, and the option to go up to node 1.0D. The display configuration is a normal decision node configuration with areas 1, 2, 3, 4, 5, and 9 active. Area 1 is labeled "NODE 1.1", area 3 is labeled "WCS", area 5 contains the "Make a mouse selection." prompt, and area 9 contains basic instructions on how to use the mouse. Area 4 contains one command, "WCS". The color code used in areas 1 and 3 is blue.

Node 1.2D. Node 1.2D is configured as a dead end node which demonstrates the use of the two dynamic menus. The main menu only supports the option to return to node 1.0D. Dynamic menu 1 (area 8) is configured for 6 selection options, and dynamic menu 2 is configured for 7 selection options. The menus are active, but a menu selection just results in a return to node 1.2D. The display configuration is a normal decision node configuration with areas 1, 2, 3, 5, and 9 active. Area 1 is labeled "NODE 1.2", area 3 is labeled "IWC", area 5 contains the "This option is not implemented (select up one level)" prompt, and area 9 contains basic instructions on how to use the mouse. The color code used in areas 1 and 3 is green.

Node 1.3D. Node 1.3D is configured as a dead end node which demonstrates a complete 20 option dynamic menu in area 8 of the screen. The main menu only supports the option to return to node 1.0D. Dynamic menu 1 is configured for 20 selection options and is active, but a menu selection just results in a return to node 1.3D. The display configuration is a normal decision node configuration with areas 1, 2, 3, 4, 5, and 9 active. Area 1 is labeled "NODE 1.3", area 3 is labeled "AIP", area 5 contains the "This option is not implemented (select up one level)" prompt, and area 9 contains basic instructions on how to use the mouse. Area 4 contains one command, "AIP". The color code used in areas 1 and 3 is red.

Node 1.4D. Node 1.4D is configured as a dead end node which demonstrates the command history and depth indicator by simulating a program level of seven. The main menu only supports the option to return to node 1.0D. The

display configuration is a normal decision node configuration with areas 1, 2, 3, 4, 5, and 9 active. Area 1 is labeled "NODE 1.4", area 3 is labeled "FILE", area 5 contains the "This option is not implemented (select up one level)" prompt, and area 9 contains basic instructions on how to use the mouse. Area 4 contains seven commands, "FILE", "COMMAND 1", "COMMAND 2", COMMAND 3", "COMMAND 4", "COMMAND 5", and "COMMAND 6". The color code used in areas 1 and 3 is yellow.

Node 1.1.1D. Node 1.1.1D is configured as a dead end node. The main menu only supports the option to return to node 1.1D. The display configuration is a normal decision node configuration with areas 1, 2, 3, 4, 5, and 9 active. Area 1 is labeled "NODE 1.1.1", area 3 is labeled "PREDRV", area 5 contains the "This option is not implemented (select up one level)" prompt, and area 9 contains basic instructions on how to use the mouse. Area 4 contains two commands, "WCS" and "PREDRV". The color code used in areas 1 and 3 is blue.

Node 1.1.2D. Node 1.1.2D is configured as the entry node into the remainder of the ROBSIM program structure. The node supports six application program choices, a help option, the configuration option, and the option to go to node 1.1D. The display configuration is a normal decision node configuration with areas 1, 2, 3, 4, 5, and 9 active. Area 1 is labeled "NODE 1.1.2", area 3 is labeled "INITDRV", area 5 contains the "Make a mouse selection." prompt, and area 9 contains basic instructions on how to use the mouse. Area 4 contains two commands, "WCS" and "INITDRV". The color code used in areas 1 and 3 is blue.

Node 1.1.3D. Node 1.1.3D is configured as a dead end node. The main menu only supports the option to return to node 1.1D. The display configuration is a normal decision node configuration with areas 1, 2, 3, 4, 5, and 9 active. Area 1 is labeled "NODE 1.1.3", area 3 is labeled "SIMDRV", area 5 contains the "This option is not implemented (select up one level)" prompt, and area 9 contains basic instructions on how to use the mouse. Area 4 contains two commands, "WCS" and "SIMDRV". The color code used in areas 1 and 3 is blue.

Node 1.1.4D. Node 1.1.4D is configured as a dead end node. The main menu only supports the option to return to node 1.1D. The display configuration is a normal decision node configuration with areas 1, 2, 3, 4, 5, and 9 active. Area 1 is labeled "NODE 1.1.4", area 3 is labeled "POSTDRV", area 5 contains the "This option is not

implemented (select up one level)" prompt, and area 9 contains basic instructions on how to use the mouse. Area 4 contains two commands, "WCS" and "POSTDRV". The color code used in areas 1 and 3 is blue.

Node 1.1.2.1D. Node 1.1.2.1D is configured as a demonstration of the interface's capability to clear the interface display from the computer screen. The main menu only supports the option of clearing the interface display from the computer screen. The display configuration is a normal decision node configuration with areas 1, 2, 3, 4, 5, and 9 active. Area 1 is labeled "NODE 1.1.2.1", area 3 is labeled "ARM", area 5 contains the "Demo to clear the GMMI off the screen (Select exit GMMI)" prompt, and area 9 contains basic instructions on how to use the mouse. Area 4 contains three commands, "WCS", "INITDRV" and "ARM". The color code used in areas 1 and 3 is blue. When the user selects the option to clear the interface display, all of the interface display except for the number 12 main menu button disappears. When the user selects button 12, the complete node 1.1.2.1D screen reappears, with only the option to return to node 1.1.2D supported.

Node 1.1.2.2D. Node 1.1.2.2 is configured as a dead end node that demonstrates both dynamic menus configured for 20 option selections. The main menu only supports the option to return to node 1.1.2D. Both dynamic menus are active, but a dynamic menu selection just results in the return to node 1.1.2.2D. The display configuration is a normal decision node configuration with areas 1, 2, 3, 4, 5, and 9 active. Area 1 is labeled "NODE 1.1.2.2", area 3 is labeled "ENVI", area 5 contains the "This is a demo of both dynamic menus. (Menus are active) (select up one level)" prompt, and area 9 contains basic instructions on how to use the mouse. Area 4 contains three commands, "WCS", "INITDRV", and "ENVIRONMENT". The color code used in areas 1 and 3 is blue.

Node 1.1.2.3D. Node 1.1.2.3D is configured as an application 1 dead end node which is at a command level of nine. The main menu only supports the option to return to node 1.1.2D. The display configuration is a normal decision node configuration with areas 1, 2, 3, 4, 5, and 9 active. Area 1 is labeled "NODE 1.1.2.3", area 3 is labeled "TARGET", area 5 contains the "This option is not implemented (select up one level)" prompt, and area 9 contains basic instructions on how to use the mouse. Area 4 contains nine commands, "WCS", "INITDRV", "TARGET", "COMMAND 4", "COMMAND 5", "COMMAND 6", "COMMAND 7", "COMMAND 8", and "COMMAND 9". The color code used in areas 1 and 3 is blue.

Node 1.1.2.4D. Node 1.1.2.4D is configured as an application 2 dead end node which is at command level 9. The main menu only supports the option to return to node 1.1.2D. The display configuration is a normal decision node configuration with areas 1, 2, 3, 4, 5, and 9 active. Area 1 is labeled "NODE 1.1.2.4", area 3 is labeled "LOAD", area 5 contains the "This option is not implemented (select up one level)" prompt, and area 9 contains basic instructions on how to use the mouse. Area 4 contains nine commands, "WCS", "INITDRV", "LOAD", and "COMMAND 4", "COMMAND 5", "COMMAND 6", "COMMAND 7", "COMMAND 8", and "COMMAND 9". The color code used in areas 1 and 3 is green.

Node 1.1.2.5D. Node 1.1.2.5D is configured as an application 3 dead end node at command level 9. The main menu only supports the option to return to node 1.1.2D. The display configuration is a normal decision node configuration with areas 1, 2, 3, 4, 5, and 9 active. Area 1 is labeled "NODE 1.1.2.5", area 3 is labeled "SYSTEM", area 5 contains the "This option is not implemented (select up one level)" prompt, and area 9 contains basic instructions on how to use the mouse. Area 4 contains nine commands, "WCS", "INITDRV", "SYSTEM", "COMMAND 4", "COMMAND 5", "COMMAND 6", "COMMAND 7", "COMMAND 8", and "COMMAND 9". The color code used in areas 1 and 3 is red.

Node 1.1.2.6D. Node 1.1.2.6D is configured as an application 4 dead end node at command level 9. The main menu only supports the option to return to node 1.1.2D. The display configuration is a normal decision node configuration with areas 1, 2, 3, 4, 5, and 9 active. Area 1 is labeled "NODE 1.1.2.6", area 3 is labeled "GRAPHICS", area 5 contains the "This option is not implemented (select up one level)" prompt, and area 9 contains basic instructions on how to use the mouse. Area 4 contains nine commands, "WCS", "INITDRV", "GRAPHICS", "COMMAND 4", "COMMAND 5", "COMMAND 6", "COMMAND 7", "COMMAND 8", and "COMMAND 9". The color code used in areas 1 and 3 is yellow.

Help nodes. Only three help nodes, corresponding to the three operationally configured decision nodes, were implemented for this thesis. The implementations of the three nodes, 1.0H, 1.1H, and 1.1.2H, are described in the following sections.

Node 1.0H. Node 1.0H is configured as the help node corresponding to decision node 1.0D. The node supports the same four application choices as node 1.0D, a

structure option, the configuration option, and the option to exit the program. The display configuration is a normal help node configuration with area 1, 2, 3, and 9 active. Area 1 is labeled "HELP 1.0", area 3 is labeled "GMMI", and area 9 contains basic instructions on how to use the mouse. Area 4-8 contains text describing the selection options supported by both decision node 1.0D and help node 1.0H. The color code displayed in areas 1 and 3 is white.

Node 1.1H. Node 1.1H is configured as the help node for the decision node 1.1D. The node supports the same four ROBSIM choices as decision node 1.0D, a structure option, the configuration option, and the option to go up to node 1.0D. The display configuration is a normal help node configuration with areas 1, 2, 3, and 9 active. Area 1 is labeled "HELP 1.1", area 3 is labeled "WCS", and area 9 contains basic instructions on how to use the mouse. Area 4-8 contains text describing the selection options supported by both decision node 1.1D and help node 1.1H. The color code used in areas 1 and 3 is blue.

Node 1.1.2H. Node 1.1.2H is configured as the help node for the decision node 1.1.2D. The node supports the same six application program choices as decision node 1.1.2D, a structure option, the configuration option, and the option to go to node 1.1D. The display configuration is a normal help node configuration with area 1, 2, 3, and 9 active. Area 1 is labeled "HELP 1.1.2", area 3 is labeled "INITDRV", and area 9 contains basic instructions on how to use the mouse. Area 4-8 contains text describing the selection options supported by both decision node 1.1.2D and help node 1.1.2H. The color code used in areas 1 and 3 is blue.

Configuration feature. The configuration feature was implemented using 12 decision nodes. The first node, do_configure, is the entry node into the configuration options. Do_file_configure, the second decision node, is a dead end node implemented to support the future addition of a capability to input a custom configuration from a disk file. The other 10 decision nodes are all part of do_custom, each allowing the user to enable or disable one screen area option, creating a customized interface configuration. Another decision node, to allow the user to save the custom configuration to a disk file, should be added to the do_custom feature in the future. The implementation of all 12 of the nodes used for the configuration option are described in the following paragraphs.

Do configure. Do_configure is the entry node into the configuration option and is configured as a standard decision node. The node supports five choices, full configuration, basic configuration, file configuration, custom configuration, and the cancel choice, which returns control to the calling node. If the full configuration option is selected, do_configure sets all of the screen area options to on, then exits back to the calling node. If the basic configuration option is selected, do_configure sets all of the screen area options to off, then exits. If the file configuration option is selected, do_configure transfers control to do_file_configuration, which is described later in this section. If the user selects custom configuration, do_configure transfers control to do_custom, also described later in this section. Screen areas 1, 2, 3, 5, and 9 are active, with area 1 containing the label "Configure 1.0" and area 3 containing the label "CONFIG". Area 5 contains the "Make a mouse selection." prompt and area 9 contains the basic instructions on how to use the mouse. The color code used in areas 1 and 3 is white.

Do file configure. Do_file_configure is a dead end node configured as a standard decision node. The node only supports the option to return to the do_configure node. Screen areas 1, 2, 3, 5, and 9 are active, with area 1 containing the label "File Config." and area 3 containing the label "CONFIG". Area 5 contains the "This option is not implemented (Select up one level)." prompt, and area 9 contains the basic instructions on how to use the mouse. The color code used in areas 1 and 3 is white.

Do custom #1. This node is the first decision node in the do_custom group of decision nodes. The node is configured as a standard decision node and supports three choice options, enable, disable, and cancel. Area 5 contains the prompt "Color on or off in the upper left area?". The selection of on enables the option, off disables the option, and the cancel choice leaves the option as is and exits the configure feature, returning control to the calling node. Screen areas 1, 2, 3, 5, and 9 are active, with area 1 containing the label "Configure 1.1" and area 3 containing the label "CONFIG". Area 9 contains the basic instructions on how to use the mouse and the color code used in areas 1 and 3 is white.

Do custom #2. This node is the second decision node in the do_custom group of decision nodes. The node is configured as a standard decision node and supports three choice options, enable, disable, and cancel. Area 5 contains the prompt "Color on or off in the upper center

area?". The selection of on enables the option, off disables the option, and the cancel choice leaves the option as is and exits the configure feature, returning control to the calling node. Screen areas 1, 2, 3, 5, and 9 are active, with area 1 containing the label "Configure 1.2" and area 3 containing the label "CONFIG". Area 9 contains the basic instructions on how to use the mouse and the color code used in areas 1 and 3 is white.

Do custom #3. This node is the third decision node in the do_custom group of decision nodes. The node is configured as a standard decision node and supports three choice options, enable, disable, and cancel. Area 5 contains the prompt "Text on or off in the upper center area?". The selection of on enables the option, off disables the option, and the cancel choice leaves the option as is and exits the configure feature, returning control to the calling node. Screen areas 1, 2, 3, 5, and 9 are active, with area 1 containing the label "Configure 1.3" and area 3 containing the label "CONFIG". Area 9 contains the basic instructions on how to use the mouse and the color code used in areas 1 and 3 is white.

Do custom #4. This node is the fourth decision node in the do_custom group of decision nodes. The node is configured as a standard decision node and supports three choice options, enable, disable, and cancel. Area 5 contains the prompt "Color on or off in the upper right area?". The selection of on enables the option, off disables the option, and the cancel choice leaves the option as is and exits the configure feature, returning control to the calling node. Screen areas 1, 2, 3, 5, and 9 are active, with area 1 containing the label "Configure 1.4" and area 3 containing the label "CONFIG". Area 9 contains the basic instructions on how to use the mouse and the color code used in areas 1 and 3 is white.

Do custom #5. This node is the fifth decision node in the do_custom group of decision nodes. The node is configured as a standard decision node and supports three choice options, enable, disable, and cancel. Area 5 contains the prompt "Text on or off in the upper right area?". The selection of on enables the option, off disables the option, and the cancel choice leaves the option as is and exits the configure feature, returning control to the calling node. Screen areas 1, 2, 3, 5, and 9 are active, with area 1 containing the label "Configure 1.5" and area 3 containing the label "CONFIG". Area 9 contains the basic instructions on how to use the mouse and the color code used in areas 1 and 3 is white.

Do custom #6. This node is the sixth decision node in the do_custom group of decision nodes. The node is configured as a standard decision node and supports three choice options, enable, disable, and cancel. Area 5 contains the prompt "Depth indication on or off in the upper right?". The selection of on enables the option, off disables the option, and the cancel choice leaves the option as is and exits the configure feature, returning control to the calling node. Screen areas 1, 2, 3, 5, and 9 are active, with area 1 containing the label "Configure 1.6" and area 3 containing the label "CONFIG". Area 9 contains the basic instructions on how to use the mouse and the color code used in areas 1 and 3 is white.

Do custom #7. This node is the seventh decision node in the do_custom group of decision nodes. The node is configured as a standard decision node and supports three options, enable, disable, and cancel. Area 5 contains the prompt "Command history on or off in the center left?". The selection of on enables the option, off disables the option, and the cancel choice leaves the option as is and exits the configure feature, returning control to the calling node. Screen areas 1, 2, 3, 5, and 9 are active, with area 1 containing the label "Configure 1.7" and area 3 containing the label "CONFIG". Area 9 contains the basic instructions on how to use the mouse and the color code used in areas 1 and 3 is white.

Do custom #8. This node is the eighth decision node in the do_custom group of decision nodes. The node is configured as a standard decision node and supports three choice options, enable, disable, and cancel. Area 5 contains the prompt "Borders on or off in the command history?". The selection of on enables the option, off disables the option, and the cancel choice leaves the option as is and exits the configure feature, returning control to the calling node. Screen areas 1, 2, 3, 5, and 9 are active, with area 1 containing the label "Configure 1.8" and area 3 containing the label "CONFIG". Area 9 contains the basic instructions on how to use the mouse and the color code used in areas 1 and 3 is white.

Do custom #9. This node is the ninth decision node in the do_custom group of decision nodes. The node is configured as a standard decision node and supports three choice options, enable, disable, and cancel. Area 5 contains the prompt "Text on or off in the command history?". The selection of on enables the option, off disables the option, and the cancel choice leaves the option as is and exits the configure feature, returning control to the calling node. Screen areas 1, 2, 3, 5, and

9 are active, with area 1 containing the label "Configure 1.9" and area 3 containing the label "CONFIG". Area 9 contains the basic instructions on how to use the mouse and the color code used in areas 1 and 3 is white.

Do custom #10. This node is the last decision node in the do_custom group of decision nodes. The node is configured as a standard decision node and supports three choice options, enable, disable, and cancel. Area 5 contains the prompt "Instructions on or off?". The selection of on enables the option, off disables the option, and the cancel choice leaves the option as is and exits the configure feature, returning control to the calling node. Screen areas 1, 2, 3, 5, and 9 are active, with area 1 containing the label "Configure 1.10" and area 3 containing the label "CONFIG". Area 9 contains the basic instructions on how to use the mouse and the color code used in areas 1 and 3 is white.

Structure feature. The structure feature of the graphical man-machine interface prototype was implemented using five structure screens and one control flow node. Descriptions of the configurations of the five screens and the control node are in the following paragraphs.

Control flow node. When the structure feature is called from a help node, the structure feature determines which structure screen (see descriptions below) is appropriate for the level and application of the calling node. The options are then set for the structure screen and it is displayed to the user. Six choices are supported by all of the structure screen configurations. The choices are; structure screen 0, structure screen 1, structure screen 2, structure screen 3, structure screen 4, and a choice to return to the calling help node. Selection of one of the structure screen results in the options being set for that screen, and it is displayed. The selection of the option to go back to help exits the structure option and returns control to the calling help node.

Structure screen 0. The configuration for structure screen 0 displays the structure of levels 0, 1, and 2 in the 4-8 area of the screen. Areas 1, 2, 3, and 9 are active, with area 1 labeled "Structure", area 3 labeled "GMMI", and area 9 contains the basic instructions on how to use the mouse. The color code used in areas 1 and 3 is white.

Structure screen 1. The configuration for structure screen 1 displays the structure of levels 1, 2, and 3 in the 4-8 area of the screen. Areas 1, 2, 3, and 9

are active, with area 1 labeled "Structure", area 3 labeled "WCS", and area 9 contains the basic instructions on how to use the mouse. The color code used in areas 1 and 3 is white.

Structure screen 2. The configuration for structure screen 2 displays a blank screen, since application 2 is not implemented, in the 4-8 area of the screen. Areas 1, 2, 3, and 9 are active, with area 1 labeled "Structure", area 3 labeled "IWC", and area 9 contains the basic instructions on how to use the mouse. The color code used in areas 1 and 3 is white.

Structure screen 3. The configuration for structure screen 3 displays a blank screen, since application 3 is not implemented, in the 4-8 area of the screen. Areas 1, 2, 3, and 9 are active, with area one labeled "Structure", area 3 labeled "AIP", and area 9 contains the basic instructions on how to use the mouse. The color code used in areas 1 and 3 is white.

Structure screen 4. The configuration for structure screen 4 displays a blank screen, since application 4 is not implemented, in the 4-8 area of the screen. Areas 1, 2, 3, and 9 are active, with area 1 labeled "Structure", area 3 labeled "FILE", and area 9 contains the basic instructions on how to use the mouse. The color code used in areas 1 and 3 is white.

Presentation Component

The presentation component consists of eight major software routines and 39 subroutines. This section is divided into eight parts, one for each of the eight major software routines and its subroutines. The major routines described in the following paragraphs are; begin_graphics, set_options, display_screen, get_user_input, clear_partial, clear_to_graphics, restore_interface, and end_graphics. Some of the subroutines are written in C, others are written in FORTRAN. The reason is that in spite of the help of many people in AFIT and at Digital Equipment Corporation, UIS graphics calls using character strings would not work from a C program on the hardware and software configuration used for this thesis effort. By the time the problem was discovered, the main part of the dialog control component was already written and running in C. To keep the thesis effort moving on schedule, all of the subroutines using character strings were written in FORTRAN and called by C routines. To indicate the language

used for a subroutine, each of the following routine names are followed by the label "C" or "FORTRAN".

Begin graphics (C). Begin_graphics just calls 12 subroutines which prepare the system for the interface graphics and draws the permanent part of the interface displays. The 12 subroutines, described in the following sections, are; set_all_colors, set_coords, open_color_maps, open_displays, graphic_colors, draw_main, draw_main_menu, draw_d_menus, draw_structure, label_area_5, label_help, and label_area_9. The subroutines called by the 12 subroutines are also described in this section.

Set all colors (C). The presentation component uses calls to the MicroVMS system graphics calls, called UIS calls. In the C programming language, UIS calls must use pointers to the values to be passed in the call. Therefore, all of the values passed in all of the UIS calls have to be set at some time before the calls are made. The set_all_colors subroutine sets all of the red, green, and blue color values which are used in later UIS calls. In this way, all of the color values used in the presentation components are in one software module, and easily changed if needed.

Set coords (C). As with the set_all_colors subroutine, the values used in the UIS drawing calls need to be set before the drawing calls are made. The set_coords subroutine sets many of those values. The values set in set_coords are mostly window sizes and locations, and the x and y locations used by line drawing routines.

Open color maps (C). The UIS graphics software requires a color map to be opened for each virtual display used by a program. The interface uses seven virtual displays and the open_color_maps subroutine opens the seven color maps.

Open displays (C). The interface uses seven virtual displays, which are opened by UIS calls in the open_display subroutine.

Graphic colors (C). The red, green, and blue values for each color in the seven color maps, opened by open_color_maps, are set by UIS graphics calls in the graphic_colors subroutine.

Draw main (C). The draw_main subroutine uses UIS graphics calls to draw all of the lines used on the main window of the interface display. The main window uses the

first virtual display. The lines consist of two types, screen outlines and area separation lines. Various shades of blue are used for the screen outlines and area separation lines. The display list for the virtual display used by the main window is disabled after the main screen features are drawn on the display. Anything drawn on the main window after draw_main is not saved by the computer and redrawn when windows to the virtual display are closed and opened.

Draw m menu (C). The draw_m_menu subroutine uses UIS graphics calls to draw all of the lines used on the main menu. The main menu uses its own virtual display, which is not shared by any other presentation feature or screen area. The virtual display is not shared with any other interface feature to allow the main menu to use its own color scheme and to separate it from other screen features during screen clears and mouse input calls. The lines drawn on the virtual display are for the menu border and the selection button outlines. The display list for the virtual display used by the main menu is disabled by draw_main_menu after the permanent features are drawn.

Draw d menus (C). The draw_d_menus subroutine uses UIS graphics calls to draw the dynamic menus on a virtual display used only by the dynamic menus. As with the draw_m_menu subroutine, the dynamic menus use their own color scheme and are independent of all other interface features. The lines drawn on the menus are just outlines of the rectangles surrounding the option areas. As with other draw subroutines, draw_d_menus disables the display list for the virtual display used by the dynamic menus after the lines are drawn on the display.

Draw structure (C). The interface structure feature also uses its own independent virtual display. The graphics that are displayed by the structure feature are drawn on the virtual display by draw_structure. Each of the different structure displays is drawn in its own section of the virtual display.

Label structure (FORTRAN). Because of the problem with the character strings in C, the structure displays were drawn with one subroutine and labeled with another. Label_structure puts the labels on the drawings made by draw_structure.

Label area 5 (FORTRAN). Label_area_5 works like draw_structure. The text for the various prompt options are put in different areas of a virtual display which is dedicated to area 5.

Label help (FORTRAN). The help function also uses a dedicated virtual display. And like label_area_5, label_help puts the text for the various help options on different parts of the help virtual display.

Label area 9 (FORTRAN). The instructions used in area 9 of the screen also use a dedicated virtual display and put different instruction labels on different parts of the display. This implementation supports multiple instruction options, even though only one instruction option is used on all of the displays implemented.

Open window (FORTRAN). The open_window subroutine is called twice by begin_graphics. It is called once to open a window, onto the virtual display for the main display area, that puts the main display in the entire computer screen area. It is called again to open a window, onto the virtual display for the main menu, that puts the main menu in area 10 of the display screen, over part of the window for the main display. The open_window subroutine is just a UIS graphics call, called from a FORTRAN routine, which opens a window to any virtual display, of any size, anywhere on the display screen.

Set options. The set_options subroutine sets all of the options for a given screen. It is called by the dialog control component before every display_screen call to set the screen options and parameters. Set_options contains a section for every screen used by the interface. Each section sets all the global variables used by all of the presentation modules for screen management and configuration control. The original design called for all of the labels used by the different display routines to be set in set_options also, but the problem with the character strings in C required another structure. If the original design had been implemented, the result would be a more efficient operation, and all of the labels and options which change from screen to screen would have been confined to one subroutine, instead of scattered throughout the program.

Get user input. Get_user_input is responsible for getting a mouse input. checking it for validity, and then passing valid mouse inputs to the calling dialog control node. Get_user_input is described in 7 sections. The first section describes the implementation of the control flow for get_user_input, and the following six sections describe 6 subroutines used by get_user_input. The six subroutines are; get_mouse_1, get_mouse_2, get_mouse_3,

mouse_input_1, mouse_input_2, mouse_input_3, and light_choice.

Get user input control flow. When called, get_user_inputs checks to see which menus are active. If just the main menu is active, get_user_input calls the subroutine get_mouse_1, which returns a value indicating a valid mouse click in the main menu window. When get_mouse_1 returns a valid mouse click, get_user_input tests it to see if it was for an active menu button. If not, get_user_input calls get mouse_1_again, and repeats the process until an input for an active button is received. Get_user_input then changes the menu button to yellow with a call to the subroutine light_choice, described later. From this point on, get_user_input handles one of two conditions. The first condition occurs when the same active button is selected two times in a row. When that occurs, get_user_input lights the selected button to red, calls display_area_2 (described later), lighting area 2 to red with the label "executing", and returns the input integer value to the calling routine. The other condition is when subsequent mouse inputs are for different active menu buttons. When this occurs, get_user_input changes the color of the first button from yellow to green, and then changes the color of the second button to yellow. This process continues until two inputs in a row are for the same active menu button.

In the case of dynamic menu 1 also being active, get_user_input calls the subroutine get_mouse_2, which checks both menus for a valid mouse click in either of the two menus. When one occurs, get_mouse_2 returns a value indicating which menu the click occurred in. From then the process is the same, except get_user_input checks both menus for the valid input.

In the case of all three menus being active, get_user_input calls the subroutine get_mouse_3, which like get_mouse_2, returns a value indicating in which menu a valid mouse click occurred.

Get mouse 1 (FORTRAN). Get_mouse_1 calls a UIS graphics asynchronous trap routine which executes mouse_input_1 when a mouse click occurs in the main menu window. The routine sets a variable to 0, then repeatedly calls the UIS routine until mouse_input_1 is executed (mouse_input_1 sets the variable to 1). Setting the variable is the signal for a down click on the mouse. Get_mouse_1 then resets the variable to 0 and loops again until mouse_input_1 again sets the variable to 1, signaling the up click on the mouse. A combined down and up click

make up one "mouse click". Get_mouse_1 then returns a one to the calling routine, signifying a mouse click in the main menu area.

Get mouse 2 (FORTRAN). Get_mouse_2 calls two UIS graphics asynchronous trap routines which execute mouse_input_1 and mouse_input_2 when a mouse click occurs in the main menu window or the dynamic 1 menu, respectively. The routine sets a variable to 0, then repeatedly calls the two UIS routines until mouse_input_1 is executed (mouse_input_1 sets the variable to 1) or mouse_input_2 is executed (mouse_input_2 sets the variable to 2). Setting the variable to a value other than 0 is the signal for a down click on the mouse in one of the two menus. Get_mouse_2 then resets the variable to 0 and loops again until mouse_input_1 or mouse_input_2 again sets the variable, signaling the up click on the mouse. Get_mouse_2 then returns a 1 or 2 to the calling routine, signifying a mouse click in the main menu area or the dynamic 1 menu area.

Get mouse 3 (FORTRAN). Get_mouse_3 calls three UIS graphics asynchronous trap routines which execute mouse_input_1, mouse_input_2, or mouse_input_3 when a mouse click occurs in the main menu window, the dynamic 1 menu, or the dynamic 2 menu, respectively. This routine sets a variable to 0, then repeatedly calls the two UIS routines until mouse_input_1 is executed (mouse_input_1 sets the variable to 1), mouse_input_2 is executed (mouse_input_2 sets the variable to 2), or mouse_input_3 is executed (mouse_input_3 sets the variable to 3). Setting the variable to a value other than 0 is the signal for a down click on the mouse in one of the three menus. Get_mouse_3 then resets the variable to 0 and loops again until mouse_input_1, mouse_input_2, or mouse_input_3 again sets the variable, signaling the up click on the mouse. Get_mouse_3 then returns a 1, 2, or 3 to the calling routine, signifying a mouse click in the main menu area, the dynamic 1 menu area, or the dynamic 2 menu area.

Mouse input 1 (FORTRAN). The mouse_input_1 routine just sets a variable to 1, when called by a UIS asynchronous trap routine.

Mouse input 2 (FORTRAN). The mouse_input_2 routine just sets a variable to 2, when called by a UIS asynchronous trap routine.

Mouse input 3 (FORTRAN). The mouse_input_3 routine just sets a variable to 3, when called by a UIS asynchronous trap routine.

Light choice (C). Light_choice lights menu status indication areas, in either the main menu or in a dynamic menu, to a color indicated by the calling routine. Light_choice either lights a choice indicator to green, yellow, red, or background blue.

Display screen. Display_screen is called after set_options at every control flow decision point in the interface. The subroutine calls, or does not call, various display area subroutines according to the screen type and screen area options. In this section, the control flow for display_screen is described first, and then each of the display area subroutine implementations are described.

Display screen control flow. When called, display_screen first calls display_area_1, then display_area_3, then display_area_9. Display_area_4a is called if the screen type is decision and area 4a is enabled. Display_area_5 is called if the screen type is decision and area 5 is enabled. Either (or both) of the dynamic menu display calls are called if they are enabled and the screen type is decision. If the screen type is help, display_area_4-8H is called, and if the screen type is structure, display_area_4-8S is called. Display_area_2, if enabled, is called last. Display_area_2 was last called by the get_user_input routine, which set the area to red with the "executing" label. By calling Display_area_2 last, display_screen is identifying the interface as ready for the next user input.

Display area 1 (C). Display_area_1 uses the application variable and the screen area options to display, or not display, the color code and node identification in area 1 of the screen. If color is enabled for area 1, display_area_1 selects the right application color and then fills in area 1. If color is not enabled, the background blue interface color is used. Text is always enabled for area 1, so display_area_1 calls label_area_1 to put the label in the area.

Label area 1 (FORTRAN). Label_area_1 uses the screen number to select the screen identification label, and then puts the label in area 1 of the display.

Display area 2 (C). When display_area_2 is called, the calling routine passes a variable indicating one of three conditions; ready, option selected, or executing. If color in area two is enabled, display_area_2 uses the variable to pick a fill color of green, yellow, or red. If color is not enabled, display_area_2 picks the background blue color as the fill color. Display_area_2

then fills in the color in area 2. If text is enabled in area 2, display_area_2 calls the FORTRAN routine label_area_2 to put the label in the area.

Label area 2 (FORTRAN). Label_area_2 uses a variable passed by the calling routine to put either "Ready", "Option selected", or "Executing" in screen area 2.

Display area 3 (C). Display_area_3 uses the application variable and the screen area options to display, or not display, the color code and application or function name in area 3 of the screen. If color is enabled for area 3, display_area_3 selects the right application color and then fills in area 3. If color is not enabled, the background blue interface color is used. If text is enabled for area 3, display_area_3 calls label_area_3 to put the label in the area.

Label area 3 (FORTRAN). Label_area_3 uses the screen number to select the application or function label, and then puts the label in area 3 of the display.

Display area 4a (C). When called, display_area_4a opens a window, showing a reserved area of the main virtual display, into area 4 of the display. Display_area_4a then uses the level variable to determine which rectangles are needed, and, if rectangles are enabled, draws them onto the display. If text is enabled, display_area_4a calls the FORTRAN subroutine label_area_4a.

Label area 4a (FORTRAN). Label_area_4a uses the screen number variable to select the command history labels, and then puts the labels in the rectangles in area 4a of the display.

Display area 4b (C). When called, display_area_4b uses the value in the dynamic_2 variable to determine how big the dynamic 2 menu should be. Display_area_4b then calls open_window to open a window onto the virtual display used for the dynamic menus, and displays the window in area 4 of the display. A call to the FORTRAN routine label_dynamic_2 is used to label the selection options. Repeated calls to light_choice are used to light the choice indication areas to green in all of the selection option areas.

Label dynamic 2 (FORTRAN). Label_dynamic_2 uses the screen number to select the selection option labels, and then puts the labels onto the dynamic 2 menu.

Display area 5 (C). Display_area_5 uses the prompt option variable to determine which part of the prompt virtual display should be shown. Display_area_5 then calls open_window to open a window onto that part of the prompt virtual display and put the prompt into area 5 of the screen.

Display area 8 (C). When called, display_area_8 uses the value in the dynamic_1 variable to determine how big the dynamic 1 menu should be. Display_area_8 then calls open_window to open a window onto the virtual display used for the dynamic menus, and displays the window in area 8 of the display. A call to the FORTRAN routine label_dynamic_1 is used to label the selection options. Repeated calls to light_choice are used to light the choice indication areas to green in all of the selection option areas.

Label dynamic 1 (FORTRAN). Label_dynamic_1 uses the screen number to select the selection option labels, and then puts the labels onto the dynamic 1 menu.

Display area 9 (C). Display_area_9 uses the instruction option variable to determine which part of the instruction virtual display should be shown. It then calls open_window to open a window onto that part of the instruction virtual display, putting the instructions into area 8 of the screen.

Display area 10 (C). Display_area_10 uses the static menu variables to set the 12 menu static menu buttons to either green or blue. For each button, display_area_10 calls light_choice, and asks for the color blue if the button is inactive, or the color green if the button is active. Display_area_10 then calls the FORTRAN routine label_area_10 to label the menu buttons.

Label area 10 (FORTRAN). Label_area_10 uses the screen number variable to select the proper menu button labels, and then puts the labels on the active menu buttons.

Display area 4-8H (C). Display_area_4-8H uses the screen number variable to determine which part of the help virtual display should be shown. It then calls open_window to open a window onto that part of the help virtual display, putting the help text into area 4-8 of the screen.

Display area 4-8S (C). Display_area_4-8S uses the screen number variable to determine which part of the

structure virtual display should be shown. It then calls open_window to open a window onto that part of the structure virtual display, putting the proper structure graphics into area 4-8 of the screen.

Clear_partial (C). Clear_partial uses the screen type variable and the screen area variables to determine what type of screen needs to be cleared and what areas in the screen are active. Areas 1,2, and 3 are always redrawn by display_screen, so clear_partial does nothing with those areas. For a decision screen, if areas 4a, 4b, 5, 8, or 9 are active, clear_partial closes the windows to the respective areas. If the screen type is a help screen, clear_partial closes the area 4-8H window and, if active, the area 9 window. The same process is followed for a structure screen with areas 4-8S and 9.

Clear_to_graphics (C). Clear_to_graphics closes the window to the main display and the window to the main menu. Then the routine resets the size of the main menu and reopens the main menu window, with a call to open_window, displaying only button 12 in the lower right corner of the screen.

Restore_interface (C). Restore_interface closes the small window to button 12 of the main window, then reopens the original windows to both the main display and the main menu, restoring the interface display.

End_graphics (C). End_graphics first closes the window to the main display and the main menu. The routine then deletes all seven of the virtual displays used by the interface.

Appendix C: Interface Evaluation

The interface evaluation consists of two parts. The first part is the evaluation cover document, which is the first four pages of this appendix. The second part of the evaluation is the CAD-Tool Human Computer Interface Evaluation, which is the last six pages of this appendix.

GRAPHICAL MAN-MACHINE INTERFACE EVALUATION

Introduction

As part of an AFIT thesis effort, a color graphical man-machine interface prototype was created as a first step towards a common interface for the robotics research software resources. In order to test the prototype and to direct future research, the interface prototype needs to be evaluated by a variety of AFIT students and staff. This project is part of that evaluation.

Evaluation Overview

The following steps are needed to successfully and accurately complete the interface evaluation:

1. Read the complete cover document. The enclosed information will improve the accuracy of the evaluation, save you time, and prevent embarrassing system crashes.

2. Thoroughly run through the interface. If you just speed through it, the evaluation results will be adversely affected.

3. Fill out the attached Interface Evaluation as accurately and as honestly as possible.

4. Return the completed evaluation to Capt Leahy or Capt Kanski, or put the evaluation in box 4251.

Thesis Description

Scope. The interface creation consisted of two parts. The first part concerned the design of a graphical man-machine interface prototype, and the second part concerned the development of the screen management techniques required to implement the interface prototype design. Both parts of the research, as well as areas not addressed by the research, are described in the following three sections.

Part one. The first part of the thesis effort consisted of a design for a graphical man-machine interface prototype which would meet basic interface requirements of the robotics software resources. The main design emphasis was on control flow, basic screen configurations, and basic man-machine interface considerations such as error handling, user skill levels, and feedback. The design did not include implementation aspects or functions required to meet specific robotics software resource requirements.

Part two. The second part of the research consisted of the development of the screen management techniques needed to successfully implement the graphical portions of the man-machine interface prototype design. The techniques were integrated with the basic control flow and displays of the design from part one to demonstrate the prototype screen management operations.

Areas not addressed. The thesis effort did not deal with any of the following man-machine interface considerations:

1. Software speed enhancements (efficiency).
2. Specific data input or output methods.
3. Data manipulations (other than data used for the screen management and specific interface displays).
4. File manipulations.
5. Color considerations in computer displays.
6. Aesthetics in color computer displays.

Interface Evaluation

The operation of the basic design and the screen management techniques will be evaluated using Dr. Hartrum's CAD-Tool Human-Computer Interface Evaluation (attached). The Interface Evaluation was developed for use with CAD programs, but the interface attributes addressed by the Interface Evaluation are valid in this context. When evaluating the interface, please keep in mind the goals

and scope of the research effort, and ignore the Interface Evaluation's emphasis on CAD tools.

Program Description

The interface program demonstrates the capability to support four robotics application programs, called WCS (ROBSIM), IWC, AIP, and FILE. Only a partial control flow structure of the ROBSIM program is implemented in the interface. The result is a limited program structure with all of the command paths coming to a dead end sooner or later. The command names used in the interface are from the real ROBSIM program, but do not execute actual ROBSIM functions.

The interface includes a variety of help functions, navigational aid functions, and interface configuration options. It shouldn't take very long to move through the implemented program structure and to try all of the supported options.

Instructions

The interface program is on two DEC GPX color workstations. The first workstation is in room 245 and is designated IVS2B. The second workstation is in room 67 and is designated RVS2A. Both workstations are running the VMS operating system. To log on to either system, click the left mouse button to activate a menu, then select the "VT220 terminal" menu option. The computer will create a window representing a VT220 terminal and will be ready for log on. Type in "ROBTEST" for the user ID and "GRAPHICS" for the password (VMS is not case sensitive).

The interface screen management functions use a lot of the available computer resources, and the speed of the interface is easily degraded by other demands on the system. Try to run the interface program when no other programs are running and when no other users are logged on to the system. Use the VMS "SHOW USERS" command to see if other users are logged on. For users experienced with the GPX workstations, have only one window open when the interface program is started. To run the interface program, type "RUN INT".

Warning: Do NOT abort the interface program with the VMS "control C" command! If you do, the graphics terminal will lock up, still showing the interface display, and the GPX workstation will have to be rebooted. The system manager also gets upset.

Use the mouse and mouse button to select menu options. The keyboard is not used by the interface program. To quit the interface program from any point, repeatedly move up the program structure until the entry level is reached, and then select "QUIT". The interface was designed to support inexperienced users, so to help test the design, no further operating instructions are given here.

After you have finished running the interface program, log out of the system with the command "LO".

When filling out the interface evaluation, please label yourself as one of the following: "instructor", "student", or "other". Also indicate your area of expertise or study as: "robotics", "graphics", or "other". You do not have to give your name. Comments concerning program bugs or research areas not addressed by this thesis effort are welcome.

As mentioned earlier, please give the completed interface evaluation to Capt Leahy or Capt Kanski, or put the evaluation in box 4251.

Thank you for your help with this thesis project.

CAD-Tool Human-Computer Interface Evaluation[1]

Name (administrative use only): _____

Estimated time spent with tool/system [:]

Do not write in these spaces
Tool Evaluated:
Class:
Group:
Exper:
First:
ID#:

PLEASE READ BEFORE PROCEEDING:

The following questionnaire is designed to provide user feedback on the human-computer interface of the specified computer-aided design (CAD) tool. Through your responses, we hope to measure your degree of satisfaction with the tool, with primary emphasis on the "user-friendliness" of the human-computer interface.

The questionnaire consists of a set of 11 factors, plus an overall rating. We will determine your satisfaction with the tool based on your response to six adjective pairs used to describe each factor. Each adjective pair has a seven-interval range where you are to indicate your feelings with an "X". Responses placed in the center of the range will indicate that you have no strong feelings one way or the other, *or that you cannot effectively evaluate that given factor.*

Evaluation begin time [:]

1. ***System Feedback or Content of the Information Displayed.*** **The extent to which the system kept you informed about what was going on in the program.**

insufficient	☐☐☐☐☐☐☐	sufficient
unclear	☐☐☐☐☐☐☐	clear
useless	☐☐☐☐☐☐☐	useful
bad	☐☐☐☐☐☐☐	good
unsatisfactory	☐☐☐☐☐☐☐	satisfactory

To me this factor is:

unimportant	☐☐☐☐☐☐☐	important

Comments:

2. *Communication.* The methods used to communicate with the tool.

complex	☐☐☐☐☐☐☐	simple
weak	☐☐☐☐☐☐☐	powerful
bad	☐☐☐☐☐☐☐	good
useless	☐☐☐☐☐☐☐	useful
unsatisfactory	☐☐☐☐☐☐☐	satisfactory

To me this factor is:

unimportant	☐☐☐☐☐☐☐	important

Comments:

3. *Error Prevention* Your perception of how well the system prevented user induced errors.

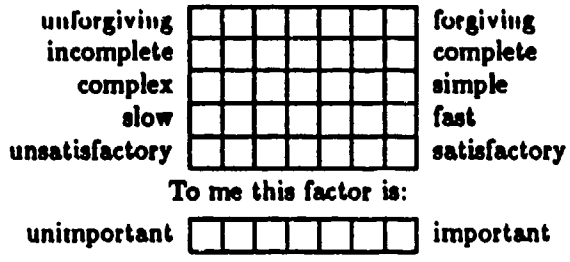
bad	☐☐☐☐☐☐☐	good
insufficient	☐☐☐☐☐☐☐	sufficient
incomplete	☐☐☐☐☐☐☐	complete
low	☐☐☐☐☐☐☐	high
unsatisfactory	☐☐☐☐☐☐☐	satisfactory

To me this factor is:

unimportant	☐☐☐☐☐☐☐	important

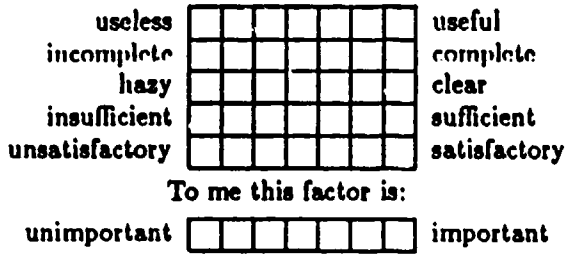
Comments:

4. *Error Recovery.* The extent and ease with which the system allowed you to recover from user induced errors.



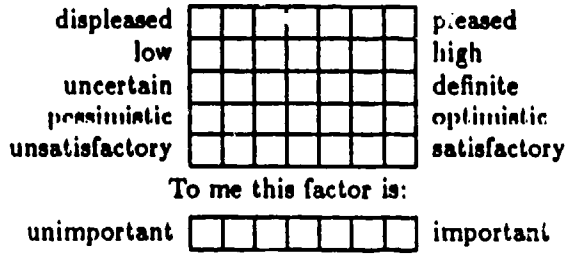
Comments:

5. *Documentation.* Your overall perception as to the usefulness of documentation.



Comments:

6. *Expectations.* Your perception as to the services provided by the system based on your expectations.



Comments:

7. ***Confidence in the System.*** **Your feelings of assurance or certainty about the services provided by the system.**

low	☐☐☐☐☐☐☐	high
weak	☐☐☐☐☐☐☐	strong
uncertain	☐☐☐☐☐☐☐	definite
bad	☐☐☐☐☐☐☐	good
unsatisfactory	☐☐☐☐☐☐☐	satisfactory

To me this factor is:

unimportant	☐☐☐☐☐☐☐	important

Comments:

8. ***Ease of Learning.*** **Ease with which you were able to learn how to use the system to perform the intended task.**

difficult	☐☐☐☐☐☐☐	easy
confusing	☐☐☐☐☐☐☐	clear
complex	☐☐☐☐☐☐☐	simple
slow	☐☐☐☐☐☐☐	fast
unsatisfactory	☐☐☐☐☐☐☐	satisfactory

To me this factor is:

unimportant	☐☐☐☐☐☐☐	important

Comments:

9. ***Display of Information.*** **The manner in which both program control and data information were displayed on the screen.**

confusing	☐☐☐☐☐☐☐	clear
cluttered	☐☐☐☐☐☐☐	well defined
incomplete	☐☐☐☐☐☐☐	complete
complex	☐☐☐☐☐☐☐	simple
unsatisfactory	☐☐☐☐☐☐☐	satisfactory

To me this factor is:

unimportant	☐☐☐☐☐☐☐	important

Comments:

10. *Feeling of Control.* Your ability to direct or control the activities performed by the tool.

low	☐☐☐☐☐☐☐	high
insufficient	☐☐☐☐☐☐☐	sufficient
vague	☐☐☐☐☐☐☐	precise
weak	☐☐☐☐☐☐☐	strong
unsatisfactory	☐☐☐☐☐☐☐	satisfactory

To me this factor is:

unimportant	☐☐☐☐☐☐☐	important

Comments:

11. *Relevancy or System Usefulness.* Your perception of how useful the system is as an aid to a software developer.

useless	☐☐☐☐☐☐☐	useful
inadequate	☐☐☐☐☐☐☐	adequate
hazy	☐☐☐☐☐☐☐	clear
insufficient	☐☐☐☐☐☐☐	sufficient
unsatisfactory	☐☐☐☐☐☐☐	satisfactory

To me this factor is:

unimportant	☐☐☐☐☐☐☐	important

Comments:

12. *Overall Evaluation of the System.* Your overall satisfaction with the system.

unsatisfied	☐☐☐☐☐☐☐	satisfied

Comments on the Overall System:

Evaluation end time [:]

Total time spent on evaluation [:]

Thank you for your help.

Appendix D: Detailed Evaluation Results

Results Description

The interface evaluation results consist of four types of data. The first type is the overall data relating to the people who filled out the evaluations. The second type of data is the numerical results for each of the interface aspects in the evaluations. The third type is comments for each of the aspects, and the last type of data is the general comments from the evaluations. The comments are labelled with a number indicating the number of the evaluation the comment came from.

Results

Table 2. Evaluator Data

Evaluator Number	AFIT Status	Area of Expertise	Time Spent	Overall Evaluation
1.	student	graphics	15	2.0
2.	student	other	17	2.0
3.	student	robotics	15	0.0
4.	instructor	other*	30	3.0
5.	instructor	other	15	2.0
6.	student	other	30	1.0
7.	student	other	15	3.0
8.	student	robotics	20	3.0
9.	student	other	21	1.0
10.	student	other	40	3.0
11.	instructor	-	15	3.0
12.	other	other	25	1.0
13.	student	graphics	15	2.0

* Man-machine interface, software engineering.

Table 3. System Feedback Results

Adjective Pair	Evaluation Number: 1	2	3	4	5	6	7	8	9	10	11	12	13	Average
1	3	2	-	3	2	3	1	2	1	3	3	1	1	2.08
2	2	2	-	3	2	2	1	3	1	3	3	1	1	2.00
3	2	2	-	0	2	2	-1	3	1	3	3	1	1	1.58
4	3	2	-	2	3	3	1	3	1	3	3	1	1	2.17
5	3	2	-	2	2	2	0	3	1	3	3	1	1	1.92
Importance	2	2	2	3	3	3	1	3	2	3	3	1	3	2.58

Average for adjective pairs (excluding importance): 1.95

System feedback comments.

(3) Tough to evaluate since the system does not execute anything.

(4) Since I don't know anything about robsim, I felt I was feeling around in the dark. I would be able to evaluate feedback better had I been familiar with the application.

(6) I like to know what it is doing, the messages were nice i.e. "executing".

(8) In "configuration", questions of yes and no nature were asked. I wasn't sure which mode I was in, in order to select a change.

(11) Use of color coordination was well done.

(12) No "lag" times; system told me that something was going on by changes in color, etc.

(13) The tool gave good info, but it was somewhat slow.

Table 4. Communication Results

Adjective Pair	Evaluation Number: 1	2	3	4	5	6	7	8	9	10	11	12	13	Average
1	3	3	3	1	3	3	3	3	3	2	3	2	3	2.69
2	2	1	0	1	3	-1	2	3	-2	2	3	2	3	1.46
3	3	1	3	1	1	2	2	3	1	3	3	2	3	2.15
4	2	1	2	2	1	2	2	3	1	3	3	2	3	2.07
5	3	2	3	2	1	2	2	3	1	3	3	2	3	2.31
Importance	2	2	1	3	0	1	3	3	2	3	3	2	3	2.15

Average for adjective pairs (excluding importance): 2.136

Communication comments.

(3) Like the mouse interface. Would be nice if executable application programs could be driven through the mouse.

(4) Menu selection is very appropriate for naive users. It would be cumbersome to the more experienced user, though.

(8) I like these types of menus better than pull down. These are easier for klutzes like me to operate.

(11) Get a new mouse that works every time you click.

(12) Good.

(13) The mouse is a good way to communicate with the tool.

Table 5. Error Prevention Results

Adjective Pair	Evaluation Number 1	2	3	4	5	6	7	8	9	10	11	12	13	Average
1	1	3	3	2	3	3	3	3	1	3	3	0	3	2.38
2	2	3	3	1	3	3	3	3	1	3	3	0	3	2.38
3	1	3	1	-1	3	0	3	3	0	3	3	0	3	1.69
4	2	3	0	1	3	-	3	3	0	3	3	0	3	2.00
5	1	3	3	1	3	2	3	3	0	3	3	0	3	2.15
Importance	3	3	3	3	3	3	3	3	0	3	3	0	3	2.54

Average for adjective pairs (excluding importance): 2.12

Error prevention comments.

(2) It appears quite "idiot resistant".

(3) Not sure I understand this evaluation grid.

(4) Menu systems typically reduce "shallow" errors. However, I think an escape to the main menu from any level of nesting would improve the system.

(6) I liked the method of having to press a block twice. For it to do something (green to yellow to red). However, I was trying to make the yellow go back to green, & it would not do this. (I found out that the yellow just moves after initially being instantiated.)

(8) I was never in a jam I couldn't recover from. Very idiot proof.

(11) Could not force interface into no win situation.

(12) Only error prevention was the message about using ^C to abort. No other errors indicated.

(13) Good error prevention.

Table 6. Error Recovery Results

Adjective Pair	Evaluation Number 1	2	3	4	5	6	7	8	9	10	11	12	13	Average
1	-1	3	3	-1	-	0	3	3	3	3	0	0	3	1.58
2	-1	3	0	3	-	0	3	3	1	3	0	0	3	1.67
3	0	3	3	-2	-	0	3	3	3	3	0	0	3	1.58
4	-1	1	-2	-2	-	0	3	3	0	3	0	0	3	.83
5	1	3	2	0	-	0	3	3	1	3	0	0	3	1.58
Importance	2	3	2	3	-	3	2	3	1	3	0	0	3	2.08

Average for adjective pairs (excluding importance): 1.45

Error recovery comments.

(4) Not handling a CRTL-C is pretty unforgiving! In the "custom" configuration function, I had to run through an exceptionally large number of questions to restore text I had inadvertently turned off. A menu selection with on/off selections would be more appropriate than user prompts.

(5) Had no errors to recover from.

(6) Never got into a situation where I needed to get back out of an error. (At least it didn't let me get to this point.)

(11) No error found to recover from.

(12) See #3.

(13) I did not crash the system.

Table 7. Documentation Results

Adjective Pair	Evaluation Number: 1	2	3	4	5	6	7	8	9	10	11	12	13	Average
1	2	2	-	-2	-	0	0	-	0	2	3	1	0	.90
2	2	2	-	0	-	0	1	-	0	2	3	1	0	1.10
3	3	2	-	-2	-	-2	0	-	0	2	3	1	1	.80
4	3	2	-	-2	-	0	0	-	0	3	3	1	1	1.10
5	2	2	-	-2	-	0	0	-	0	3	3	1	1	1.00
Importance	1	2	2	3	-	1	2	-	2	3	3	1	3	2.09

Average for adjective pairs (excluding importance): .98

Documentation comments.

(2) It appears as though "Instructions #1" at lower left screen is erased and rewritten w/each screen change- time might be saved by not doing this.

(3) No documentation to evaluate.

(4) I presume by "documentation" you mean on-line help? Context sensitive help would have added significantly to your software.

(5) N/A

(6) I would have liked to have help available when you had screens of 52 selections, i.e. interactive help to let you know what each selection is.

(7) Help text: Too much for just one screen. I would've liked it more if help text was available on other screens.

(8) Help function extremely well laid out.

(9) N/A

(11) Login procedures only documented & worked perfectly.

(13) Fairly good documentation.

Table 8. Expectations Results

Adjective Pair	Evaluation Number: 1	2	3	4	5	6	7	8	9	10	11	12	13	Average
1	2	2	1	0	-	0	2	3	-1	2	3	1	2	1.41
2	1	2	1	0	-	0	2	3	0	1	3	1	2	1.33
3	1	2	0	0	-	0	1	3	1	1	3	1	1	1.17
4	1	2	2	0	-	0	2	3	0	2	3	1	2	1.50
5	2	2	1	0	-	0	2	3	0	3	3	1	2	1.58
Importance	2	2	0	0	-	-2	2	3	0	1	3	1	2	1.17

Average for adjective pairs (excluding importance): 1.39

Expectations comments.

(3) I wish there was an actual piece of application software to execute. This would give me a better feel for how I liked the interface.

(4) Response time (even on an unloaded system) was disappointing. Yes, I realize this is a prototype, but golly, response time got very old.

(5) None expected; None delivered.

(6) I didn't know what to expect. It seems ok.

(11) No problems!

Table 9. Confidence in the System Results

Adjective Pair	Evaluation Number: 1	2	3	4	5	6	7	8	9	10	11	12	13	Average
1	3	2	2	0	2	1	-	3	1	3	3	1	2	1.91
2	2	2	2	0	2	1	-	3	1	3	3	1	3	1.92
3	2	2	1	0	2	1	-	3	1	2	3	1	2	1.67
4	2	2	2	0	2	1	-	3	1	3	3	1	2	1.83
5	2	2	1	0	2	1	-	3	1	3	3	1	3	1.83
Importance	1	3	2	3	0	2	-	3	1	3	3	1	3	2.08

Average for adjective pairs (excluding importance): 1.83

Confidence in the system comments.

(4) Neutral on this one.

(6) It seems ok.

(8) It worked!

(9) I'm not sure I understand the question.

(11) Easy to get through all menus accurately.

Table 10. Ease of Learning Results

Adjective Pair	Evaluation Number: 1	2	3	4	5	6	7	8	9	10	11	12	13	Average
1	3	3	3	-	3	3	-	3	-1	3	3	2	3	2.54
2	3	3	2	-	3	2	-	3	1	3	3	2	3	2.54
3	3	3	2	-	3	3	-	3	2	3	3	2	3	2.73
4	1	3	3	-	3	3	-	3	0	3	3	2	3	2.45
5	2	3	2	-	3	3	-	3	1	3	3	2	3	2.54
Importance	3	3	3	-	3	1	-	3	1	3	3	2	3	2.54

Average for adjective pairs (excluding importance): 2.56

Ease of learning comments.

(4) What task was I performing?

(8) Jumped on and flew!

(9) I missed the message that said to double-click for about 5 minutes, then I realized my error.

(11) Simple flow.

(12) Twice through, it became quite easy. Pop-up menus, (or menu nesting) is very easy to use.

(13) Easy to use.

Table 11. Display of Information Results

Adjective Pair	Evaluation Number: 1	2	3	4	5	6	7	8	9	10	11	12	13	Average
1	3	2	3	3	2	2	-	3	1	3	3	2	3	2.50
2	2	2	3	3	0	2	-	3	1	2	3	2	3	2.16
3	2	2	3	3	0	-1	-	3	1	2	3	2	3	1.92
4	2	2	3	3	2	2	-	3	1	3	3	2	3	2.42
5	2	2	3	3	1	2	-	3	1	3	3	2	3	2.33
Importance	2	3	2	3	2	1	-	3	1	3	3	2	3	2.33

Average for adjective pairs (excluding importance): 2.27

Display of information comments.

(3) Excellent layout. Avoids large mouse movements, keeps users attention focused at eye level.

(8) I really like the segmented screen rather than overlayed screens. I can't remember underlying screens. Here I can see everything at once.

(11) Nice big choice boxes & display areas.

(12) I thought the physical layout was good. The system is "right handed" (menu controls are located lower right). For me that's ok, but would it make a difference to leftys?

Table 12. Feeling of Control Results

Adjective Pair	Evaluation Number: 1	2	3	4	5	6	7	8	9	10	11	12	13	Average
1	2	3	3	3	3	3	3	3	3	3	3	1	3	2.77
2	2	3	3	3	3	3	3	3	2	3	3	1	3	2.69
3	3	3	3	3	3	3	1	3	2	3	3	1	3	2.61
4	2	3	3	3	3	3	3	3	2	3	3	1	3	2.69
5	2	3	3	3	3	3	2	3	2	3	3	1	3	2.61
Importance	0	3	3	1	3	1	2	3	1	3	3	1	3	2.07

Average for adjective pairs (excluding importance): 2.67

Feeling of control comments.

(3) Another excellent area.

(6) I felt like I had control over every step. The system did not seem intimidating to use.

(8) I never had the feeling I wasn't in control. Truly successful brainwashing takes place when the machine makes you believe you know what you're doing!

(12) Ok

Table 13. Relevancy or System Usefulness Results

Adjective Pair	Evaluation Number: 1	2	3	4	5	6	7	8	9	10	11	12	13	Average
1	0	3	0	-	-	1	3	3	0	3	3	1	2	1.72
2	0	3	0	-	-	1	3	3	0	3	3	1	1	1.63
3	1	3	-3	-	-	1	3	3	0	3	3	1	1	1.45
4	0	3	0	-	-	1	0	3	0	3	3	1	2	1.45
5	0	3	0	-	-	1	2	3	0	3	3	1	2	1.64
Importance	2	3	1	-	-	-2	3	3	0	3	3	1	3	1.82

Average for adjective pairs (excluding importance): 1.58

Relevancy or system usefulness comments.

(3) Is this meant for developing applications or executing them?

(4) N/A

(5) N/A

(9) N/A

(11) Appears good.

Table 14. Overall Evaluation of the System

Adjective Pair	Evaluation Number: 1	2	3	4	5	6	7	8	9	10	11	12	13	Average
1	2	2	0	3	2	1	3	3	1	3	3	1	2	2.0

(4) For a bare-bones prototype.

(11) Nice Job.

General comments.

(1) This may be of some help to a software developer, but would be of much greater use to a user. I was surprised by getting pushed out of the system the first time I went into help at the high level. This occurred twice.*

(2) Not bad. Initial expectations were that it would be faster.

(3) This thing is slow! In fact I might be tempted to avoid using it because it is so slow (if I was performing a task requiring lots of commands to the interface, speed would be a real problem). Really liked the mouse, layout (push buttons), structure trees, and use of color. I don't feel the custom configuration option adds anything to the system (the full configuration is fine for me). Added note: This is one of the stranger evaluation forms I've ever filled out. I'm not entirely sure I understand the intent or meaning of all the adjective pairs (some seem vague or redundant redundant.

(4) It is difficult to evaluate an interface to an unimplemented application. Real use is where the rubber meets the road. I can only look at the prototype in terms of how aesthetically pleasing it is - which is important, but not critical to use in a working environment.

(6) I think the system was pretty good. Especially good was error prevention, simplicity, system feedback, & ease of learning. Shortfalls were; online documentation for some functions, & limitations of a mouse. (I am assuming that there may be reasons why you would need to have data input from the keyboard, & that you are not implementing this.)

(7) Need an on screen indication other than just a color change to indicate a second button press is needed to confirm the choice. How to abort a selected option is not obvious to the user.

(8) Very clear and understandable. A pleasure to use, not a chore.

(9) Some of the lower nodes will show a series of boxed commands. In addition to the implemented commands, there are boxes for "command 4" . . "command 9". If these aren't going to have "real" commands, why show them? Why do you require double clicks? Especially with the problems the system has recognizing just a single click. I think the only option that should require a double click is quit. Why not allow the user to quit from any level, instead of having to trace back to the top menu? I don't understand the use of different colors in the top bar (the two bars beside the "option selected/executing" section). Why not keep this bar the same color (blue) always, or correlate it with the color of the menu item (i.e. have robsim menu item be orange, and all sub menu items be orange, along with the top bar? The large font in the top bar ("Node 1.2", etc.) looks pretty bad. Aren't there any smoother-looking fonts to use? The "dynamic menus" are simple to use. But why are they numbered 13-32 on the RIGHT side, and 33-52 on the LEFT side. This is counter-intuitive.

(10) I LIKE the "Up one level" command. Too often a new user heads down a "path" w/no idea as to how to extricate himself other than to shut the whole darn thing off & start over. Along this line (screen space permitting) a "map" of where the user is in the command structure would be helpful (see sample).** Such a window could be on the screen all the time, or easily called up.

(11) Clicking only perceived problem.

(12) Only element I did not like was having to click the mouse twice for a menu selection. Most interfaces I am used to have you click only once to select. If someone used this for hours and hours it may become cumbersome. Problems sometimes were caused by clicking the mouse too quickly.

* The failure could not be reproduced.

** The evaluator included a drawing that looked exactly like a structure screen display, except it included a "you are here" indicator.

Bibliography

1. Borenstein, Nathaniel Solomon. "The Design of On-Line Help Systems," (Abstract) SIGCHI Bulletin, 18: 24-25 (July 1986).

2. Borning, Alan and Robert Duisberg. "Constant-Based Tools for Building User Interfaces," ACM Transactions on Graphics, 5: 345-374 (October 1986).

3. Britts, Stefan. "Dialog Management in Interactive Systems: A Comparative Survey," SIGCHI Bulletin, 18: 30-39 (January 1987).

4. Butcher, Michael D. "A Model For An Interactive Graphical Interface To An Electronic Office System," (Abstract) SIGCHI Bulletin, 18: 66-68 (October 1986).

5. Cardelli, Luca and Rob Pike. "Squeak: A Language for Communicating with Mice," ACM SIGGRAPH Computer Graphics, 19: 199-204 (July 1985).

6. Foley, J. D. and A. Van Dam. Fundamentals of Interactive Computer Graphics. Reading, Massachusetts: Addison-Wesley Publishing Company, 1984.

7. Giloth, Copper and Jane Veeder. "The Paint Problem," IEEE Computer Graphics & Applications, 5: 66-74 (July 1985).

8. Green, Mark. "The University of Alberta User Interface Management System," ACM SIGGRAPH Computer Graphics, 19: 205-213 (July 1985).

9. Grimm, Sigrid and others. "A User Needs Approach To Context-Sensitive Help," SIGCHI Bulletin, 19: 65-66 (January 1988).

10. Hartrum, Thomas C. CAD-Tool Human-Computer Interface Evaluation. Air Force Institute of Technology (ENG), Wright-Patterson AFB OH, 22 April 1988.

11. Hill, Ralph D. "Supporting Concurrency, Communication, and Synchronization in Human-Computer Interaction - The Sassafras UIMS," ACM Transactions on Graphics, 5: 179-210 (July 1986).

12. Holynski, Marek and others. "An Adaptive Graphics Analyzer as a Preference-Oriented Interface," (Abstract) SIGCHI Bulletin, 19: 46-48 (October 1987).

13. Horton, Marjorie S. "Helping Users Find Help: Models of Online Help Organization," SIGCHI Bulletin, 18: 48-49 (October 1986).

14. Isa, Barbara S. and others. "Navigation Issues Related To Menu-Based Software Products," SIGCHI Bulletin, 18: 68-69 (October 1986).

15. Kalay, Yehuda E. "Worldview: An Integrated Geometric-Modeling Drafting System," IEEE Computer Graphics & Applications, 7: 36-46 (February 1987).

16. Kanski, Jerry J. Graphical Man-Machine Interface Prototype Program Listings, Version 1.0. AFIT Robotic Systems Laboratory Technical Report AFSR-88-1. Air Force Institute of Technology (ENG), Wright-Patterson AFB OH, December 1988.

17. Kuhlmann, Herbert W. and Bernd W. Alheit. "Communicating Via Graphics Standards Interfaces," Computers & Graphics, 10: 97-102 (1986).

18. Lantz, Keith A. "On User Interface Reference Models," SIGCHI Bulletin, 18: 36-42 (October 1986).

19. Leach, Ronald J. "Evaluating the Performance of a User Interface," Computers & Graphics, 11: 141-146 (1987).

20. ----- "Graphical Control Systems and Multiple Displays," Computers & Graphics, 9: 415-422 (1985).

21. Maguire, M. C. "A Review of Human Factors Guidelines and Techniques for the Design of Graphical Human-Computer Interfaces," Computers & Graphics, 9: 221-235 (1985).

22. Marcus, Aaron and others. "An Icon Design Manual Page for the User Interface of a CAD/CAM/CAE Workstation," (Abstract) SIGCHI Bulletin, 19: 55 (October 1987).

23. Mittelman, Jeffrey P. "The ARF Workstation: A New Application of Human-Computer Interface Principles," Computers & Graphics, 11: 147-156 (1987).

24. Morgan, Chris. "An Interview with Wayne Rosing, Bruce Daniels, and Larry Tesler," BYTE, 8: 90-114 (February 1983).

25. Myers, Brad A. "Creating Interaction Techniques by Demonstration," IEEE Computer Graphics & Applications, 7: 51-60 (September 1987).

26. Myers, Brad A. and William Buxton. "Creating Highly-Interactive and Graphical User Interfaces by Demonstration," ACM SIGGRAPH Computer Graphics, 20: 249-257 (August 1986).

27. Myers, Ware. "Computer Graphics: The Next 20 Years," IEEE Computer Graphics & Applications, 5: 69-76 (August 1985)

28. National Aeronautics and Space Administration. Evaluation of Automated Decisionmaking Methodologies and Development of an Integrated Robotic System Simulation. NASA Contractor Report 178050. Langley Research Center, Hampton, Virginia, March 1986.

29. National Aeronautics and Space Administration. Evaluation of Automated Decisionmaking Methodologies and Development of an Integrated Robotic System Simulation, Appendix A - ROBSIM User's Guide. NASA Contractor Report 178051. Langley Research Center, Hampton, Virginia, March 1986.

30. Newman, William M. and Robert F. Sproull. Principles of Interactive Computer Graphics. New York: McGraw-Hill Book Company, 1979.

31. Nielsen, Jakob. "Classification of Dialog Techniques: A CHI+GI Workshop, Toronto, April 6, 1987," SIGCHI Bulletin, 19: 30-35 (October 1987).

32. Pressman, Roger S. Software Engineering: A Practitioner's Approach. New York: McGraw-Hill Book Company, 1987.

33. Richer, Mark H. and William J. Clancy. "GUIDON-WATCH: A Graphic Interface for Viewing a Knowledge-Based System," IEEE Computer Graphics & Applications, 5: 51-63 (November 1985).

34. Scheifler, Robert W. and Jim Gettys. "The X Window System," ACM Transactions on Graphics, 5: 79-109 (April 1986).

35. Sibert, John L. and others. "An Object-Oriented User Interface Management System" ACM SIGGRAPH Computer Graphics, 20: 259-267 (August 1986).

36. Tanner, Peter P.and others. "A Multitasking Switchboard Approach to User Interface Management," ACM SIGGRAPH Computer Graphics, 20: 241-248 (August 1986).

37. Taylor, Roderick Austin. "Using Multiple Dialog Modes in a User-System Interface to Accomodate different Levels of User Experience: An Experimental Study," (Abstract) SIGCHI Bulletin, 19: 12 (October 1987).

38. Thimbleby, Harold. "Failure In The Technical User-Interface Design Process," Computers & Graphics, 9: 187-193 (1985).

39. Tyler, Sherman William. "SAUCI: Self-Adaptive User-Computer Interface," (Abstract) SIGCHI Bulletin, 19: 11-12 (October 1987).

40. Weber, Helmut Richard. "Meditation on Man-Machine Interfaces or Our Personal Role in Graphics Dialogue Programming," Computers & Graphics, 9: 237-245 (1985)

41. Williams, Antony. "An Architecture for User Interface R&D," IEEE Computer Graphics & Applications, 6: 39-49 (July 1986).

42. Williams, Gregg. "The Apple Macintosh Computer," BYTE, 9: 30-54 (February 1984).

43. ----- "The Lisa Computer System," BYTE, 8: 33-50 (February 1983).

VITA

Captain Jerry J. Kanski was born on 9 January 1954 in Billings, Montana. He graduated from high school in Billings in 1972. He attended Montana State University from 1972 to 1975. In 1980, he enlisted in the USAF and served as a flight test engineering assistant at the Air Force Flight Test Center, Edwards AFB, California, until 1982. From 1982 to 1984 he again attended Montana State University, where he received the degree of Bachelor of Science in Electrical Engineering. Upon graduation, he received a commission through OTS. He served as a computer systems engineer at the Air Force Weapons Laboratory, Kirtland AFB, New Mexico, from 1984 until June 1987, when he entered the School of Engineering, Air Force Institute of Technology.

Permanent address: 1022 St. Johns Ave.
Billings, Montana 59102

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) AFIT/GE/ENG/88D-17	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION School of Engineering 6b. OFFICE SYMBOL (If applicable) AFIT/ENG	7a. NAME OF MONITORING ORGANIZATION
6c. ADDRESS (City, State, and ZIP Code) Air Force Institute of Technology Wright-Patterson AFB OH 45433-6583	7b. ADDRESS (City, State, and ZIP Code)
8a. NAME OF FUNDING/SPONSORING ORGANIZATION Armstrong Aero. Med. Research Laboratory 8b. OFFICE SYMBOL (If applicable) AAMRL/BBA	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State, and ZIP Code) Wright Patterson AFB OH 45433-6583	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT ACCESSION NO.

11. TITLE (Include Security Classification)
GRAPHICAL MAN-MACHINE INTERFACE FOR AN INTEGRATED EVALUATION ENVIRONMENT

12. PERSONAL AUTHOR(S)
Jerry J. Kanski, B.S., Captain, USAF

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
MS Thesis	FROM _____ TO _____	1988 December	161

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	
12	05		Computer Graphics
23	02		Man Computer Interface

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

Thesis Advisor: Michael B. Leahy Jr., Captain, USAF

ASST PROF OF EE

Approved for public release: IAW AFR 190-1
10 Jan 89

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT	21. ABSTRACT SECURITY CLASSIFICATION
☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS	UNCLASSIFIED
22a. NAME OF RESPONSIBLE INDIVIDUAL Michael B. Leahy Jr., Captain, USAF	22b. TELEPHONE (Include Area Code) (513)255-6027
	22c. OFFICE SYMBOL AFIT/ENG

DD Form 1473, JUN 86 — Previous editions are obsolete. — SECURITY CLASSIFICATION OF THIS PAGE

UNCLASSIFIED

The purpose of this research was to create an effective color graphical man-machine interface prototype for the various software resources used by AFIT researchers. The research was completed in four steps: (1) The general user requirements and specific interface requirements for the interface were determined. (2) An interface design which met all of the requirements determined in step 1 was created. (3) The design created in step 2 was implemented. (4) The man-machine interface was evaluated by a group of representative users.

The design effort resulted in a general purpose interface design that was highly adaptable and expandable. The design met all of the requirements determined in step 1, yet included the flexibility to meet future, as yet undetermined, requirements. The implementation of the interface design also featured adaptability and flexibility. The implementation combined a variety of graphical techniques to create a very powerful system. In addition, the resulting man-machine interface was evaluated by a group of AFIT engineering students and staff. The results of the evaluation indicated that the graphical man-machine interface prototype was a success.